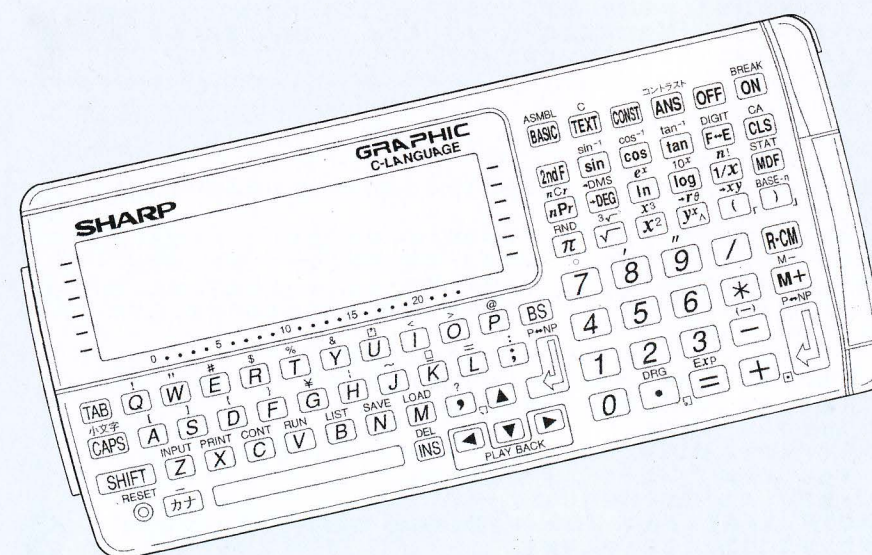
SHARP

取扱説明書

ポケットコンピュータ

形名 PC-G850VS

保証書付（巻末）
(WITH WARRANTY CARD)

ページ

安全にお使いいただくために

図記号について この取扱説明書には、安全にお使いいただくためにいろいろな表示をしています。その表示を無視して誤った取り扱いをすることによって生じる内容を、次のように区分しています。
内容をよく理解してから本文をお読みになり、記載事項をお守りください。

⚠ 警告	人が死亡または重傷を負うおそれがある内容を示しています。
⚠ 注意	人がけがをしたり財産に損害を受けるおそれがある内容を示しています。

図記号の意味
⚠ 記号は、気をつける必要があることを表しています。
⃠ 記号は、してはいけないことを表しています。
❗ 記号は、しなければならないことを表しています。

⚠ 警　告

- この製品およびＡＣアダプターの上やそばに水やコーヒーなど液体の入った容器を置かないでください。倒れて内部に水などが入ると、火災や感電の原因となります。
- お客様による改造や修理はしないでください。火災や感電、けがの原因となります。

- この製品には、ＥＡ－23Ｅ以外のＡＣアダプターは使用しないでください。他のＡＣアダプターをご使用になると、故障や火災の原因となります。

- ＡＣアダプターの電源はＡＣ100Ｖ（50／60Hz）のコンセントを使用してください。それ以外の電源で使用されますと、火災の原因となります。
- ＡＣアダプターをコンセントに直接接続してください。タコ足配線は加熱し、火災の原因となります。
- ぬれた手でＡＣアダプターを抜き差ししないでください。感電の原因となります。
- 電源コードを傷つけたり、破損したり、加工したりしないでください。また重いものを載せたり、引っ張ったり、無理に曲げたりすると電源コードを傷め、火災や感電の原因となります。
- 万一、発熱していたり、煙が出ている、変な臭いがするなどの異常状態のままで使用しますと、火災、感電の原因となります。すぐにこの製品の電源スイッチを切り、ＡＣアダプターをコンセントから抜き、お買いあげの販売店にご連絡ください。

- 雷がなりはじめたら、落雷による感電・火災の防止のため、この製品の電源を切り、ＡＣアダプターをコンセントから抜いてください。

⚠ 注　意

- 使用しないときはＡＣアダプターをコンセントおよびＡＣアダプター接続端子から外しておいてください。
- この製品およびＡＣアダプターは、ほこりや湿気の多い場所で使用しないでください。ほこりや汚れ・水滴がつきますと、火災や感電・漏電の原因となることがあります。
- 電池は誤った使いかたをすると、破れつや発火の原因となることがあります。また、液もれして機器を腐食させたり、手や衣服などを汚す原因となることもあります。以下のことをお守りください。
  - ・プラス⊕とマイナス⊖の向きを表示どおり正しく入れる。
  - ・種類の違うものや新しいものと古いものを混ぜて使用しない。
  - ・使えなくなった電池を機器の中に放置しない。
  - ・端子をショートさせない。
  - ・水や火の中に入れたり、分解しない。
  - ・もれた液が体についたときは、水でよく洗い流す。
  - ・充電池（ニカド電池）は使用しない。
- ＡＣアダプターを抜くときは、電源コードを引っ張らないでください。コードが傷つき、火災、感電の原因となることがあります。

＜はじめに＞

お買いあげいただき、まことにありがとうございました。
この取扱説明書をよくお読みのうえ、正しくお使いください。
ご使用の前に、「安全にお使いいただくために」を必ずお読みください。
この取扱説明書は、いつでも見ることができる場所に必ず保存してください。

本書で使用する用語は、情報処理技術者用の一般書籍にできるだけ準じています。

〈ご使用前のおことわり〉

- お客様または第三者がこの製品および付属品の使用を誤ったことにより生じた故障、不具合、またはそれに基づく損害については、法令上の賠償責任が認められる場合を除き、当社は一切その責任を負いませんので、あらかじめご了承ください。
- この製品は付属品を含め、改良のため予告なく変更することがあります。

〈おことわり〉

本書に記載の別売周辺機器（CE-126P（プリンタ）、CE-T800（パソコン通信ケーブル）、EA-129C（ポケコン接続ケーブル））は2007年12月現在、販売を完了しております。
あらかじめご了承ください。
すでに上記周辺機器をお持ちの場合は、この製品に接続してお使いいただけます。

〈記憶内容保存のお願い〉

故障・修理のときや電池交換を行ったときは、記憶内容が消失します。また、この製品の使いかたを誤ったときや静電気・電気的ノイズの影響を受けたときは、記憶内容が変化・消失する恐れがあります。
重要な内容は必ず紙などに控えておいてください。

この取扱説明書の内容にあたっては、新潟県下の工業高校（新潟県立長岡工業高等学校、新潟県立三条工業高等学校、新潟県立新潟工業高等学校、新潟県立巻工業高等学校、新潟県立高田工業高等学校、新潟県立小千谷西高等学校）ならびに岐阜県立岐阜工業高等学校の先生方、および関係者の方々に多大なご協力をいただきました。
この場をお借りし、心から感謝申しあげます。　（平成13年７月現在）

もくじ

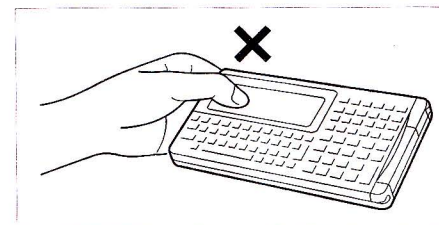
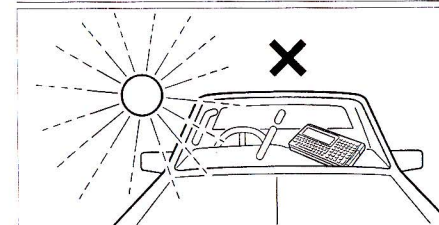
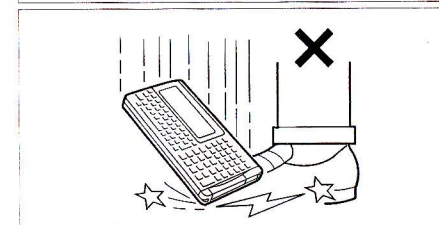
第8章 機械語モニタとアセンブラ機能 ……273

第9章 PIC ……306

第10章 BASICの各命令の説明 ……315

本機と周辺機器（プリンタなど）接続時のご注意
● **本機の電源を切ってから、周辺機器と接続してください。**
電源を入れたままで接続すると、故障の原因となります。

お使いになる前に

1．おねがい

液晶表示部を強く押さえないでください。
表示部はガラスでできていますので、押さえると割れることがあります。

日の当たる自動車内・直射日光が当たる場所・暖房器具の近くなどに置かないでください。
高温により、変形や故障の原因になります。

落としたり、ぶつけたり、強いショックを与えないでください。
大きな力が加わり、壊れることがあります。

お手入れは、乾いたやわらかい布で軽くふいてください。シンナーやベンジンなど、揮発性の液体やぬれた布は使用しないでください。変質したり色が変わったりすることがあります。

本体裏面に貼っているネームラベルにお名前をご記入ください

● **カバンには硬いものや先のとがった物と一緒に入れないでください。**傷がつくことがあります。
また、ハードカバーを必ず取り付けてください。
● **防水構造になっていませんので、水など液体がかかるところでの使用や保存は避けてください。**
雨、水しぶき、ジュース、コーヒー、蒸気、汗なども故障の原因となります。
● **本機に周辺機器（プリンタなど）を接続するときは、本機の電源を切ってから行ってください。**
電源を入れたままで接続すると、故障の原因となります。

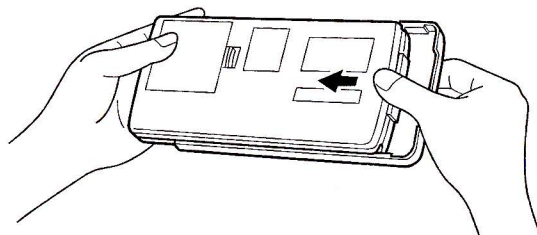
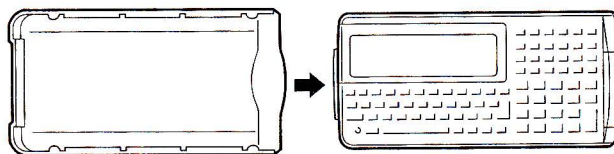
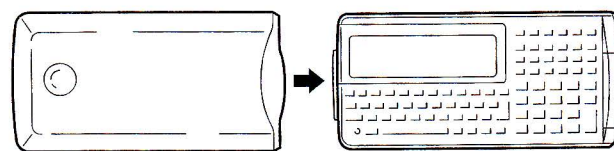
この装置は、情報処理装置等電波障害自主規制協議会（ＶＣＣＩ）の基準に基づくクラスＢ情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。
取扱説明書に従って正しい取り扱いをしてください。

正しい取り扱いをしても、電波の状況によりラジオ、テレビジョン受信機の受信に影響を及ぼすことがあります。そのようなときは、次の点にご注意ください。

- この製品をラジオ、テレビジョン受信機から十分に離してください。
- ＡＣアダプターとラジオ、テレビジョン受信機を別のコンセントに接続してください。
- 使用されるケーブルは指定のものを使用してください。

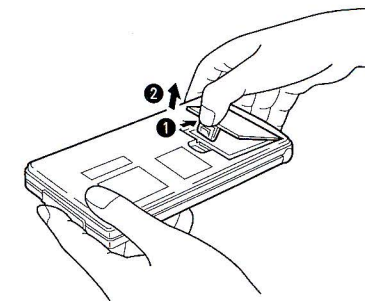
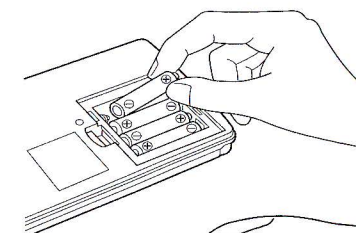
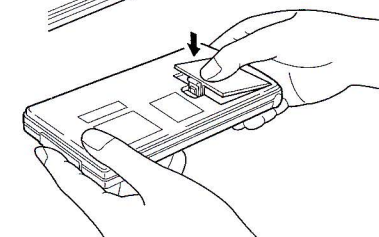
ハードカバーの使いかた

ハードカバーは衝撃に対して本体を保護する役割を持っています。
ポケットコンピューターを使用しないとき、また、カバンに入れて持ち運ぶときなどにはハードカバーを取り付けてください。

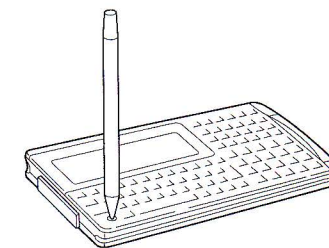
ハードカバーの取り外しかた

使うとき

使わないとき

2．お買いあげ後はじめてご使用になるときの操作

(1) 電池の取り付け

付属の乾電池を次の手順で入れてください。

①本体裏面の電池ぶたを図のようにして外します。

②乾電池をマイナス⊖側から入れます。⊕・⊖をまちがえないように入れてください。

③電池ぶたをもとどおり取り付けます。

(2) 初期設定

電池を入れた直後は、本機の内部が不安定な状態になっています。次の操作で初期設定を行ってください。初期設定をしないでいると、電池が早く消耗します。

① ON キーを押して電源を入れた後、ボールペンなどで、本体左端のリセット（ＲＥＳＥＴ）スイッチを押してください。
芯先の出たシャープペンシルや先の折れやすいもの、また、針など先のとがったものは使用しないでください。

②リセットスイッチを離すと次の画面になります。違う画面になったときはもう一度リセットスイッチを押してください。

MEMORY CLEAR O. K. ? (Y/N)

(メモリー内容を消去しますか?)

本書の表示例では、説明上必要なシンボルだけを記載しています。(シンボルについては22ページを参照してください。)

③Y キーを押してください。次の画面(点滅)になります。
(初期設定し、記憶内容をすべて消去したことを示しています。)

④↵ キーを押してください。次の画面になります。

メモ キーの表記について

説明中の ON や Y は、それぞれのキーを押すことを示しています。
最初は説明どおりにキー操作を行い、ポケットコンピュータになれましょう。

(3) 動作確認

正常に動作しているか確認するために、次のキー操作を行ってください。
F R E ↵

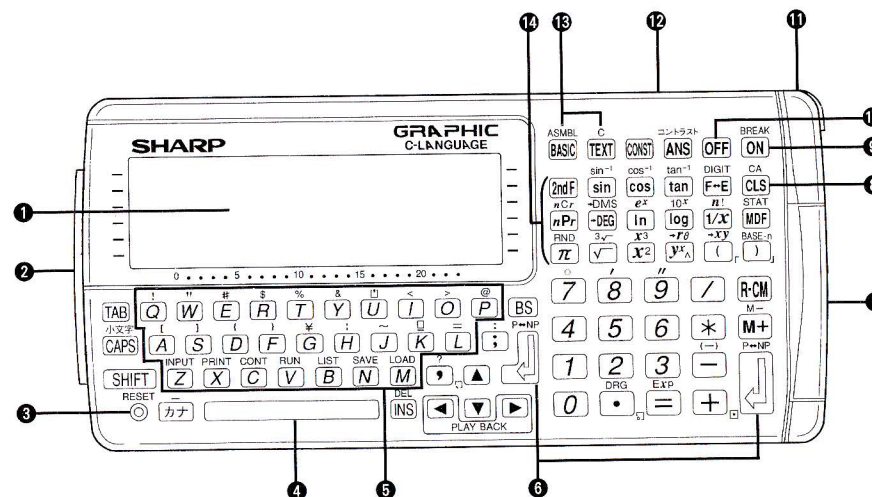
RUN MODE
FRE
30179.

以上のように表示すれば本機は正常に動作しており、指令待ち状態になっています。
最後に表示された"30179."という値は、本機にプログラムやデータを入れることができる最大の容量を示しています。
(注) (2)〜(3)の操作を行っても説明どおりにならない場合は、もう一度読み直して、操作をやり直してください。

電池交換について

"BATT"シンボルが画面左下に点灯したときは、本機内の電池が消耗したことを示しています。このシンボルが点灯したときは、一度 OFF で電源を切り、ON で再度電源を入れてください。それでもなおこのシンボルが消えないときは、速やかに372ページの手順に従って電池を交換してください。
このシンボルが点灯したままで使っていると、突然電源が切れたり、電源が入らなくなったりします。

(注)・電池を交換するとプログラムやデータが消えます。
電池を交換する前にプログラムやデータを紙に書き写しておいてください。周辺機器CE-126P(プリンタ)をお持ちの場合は印字しておいてください。
・別売のACアダプターEA-23Eをご使用の場合も、ACアダプターを抜いて、本機内の電池が消耗していないか確かめてください。
電池が消耗していると、ACアダプターを取り外したときにプログラムやデータが消えてしまいます。

3. 各部のなまえ

❶ 表示部
❷ 周辺機器接続端子(11ピン)
❸ リセットスイッチ
❹ スペースキー
❺ アルファベットキー
❻ キャリッジリターンキー
❼ システムバス端子
❽ クリア/クリアオールキー
❾ 電源オン/ブレークキー
❿ 電源オフキー
⓫ ACアダプター接続端子
⓬ 電池ぶた(裏面)
⓭ モード切り替えキー
⓮ 関数キー

(注) ● キャリッジリターンキーはどちらのキーを押しても同じです。
● スペースキー"[ ]"は本書では SPACE と記載します。
● ACアダプター(EA-23E)は別売品です。

基本操作とモードについて

本機には多くのキーがあります。各キーにはいろいろな文字や数字、記号が書かれています。
ここでは、これらの操作のしかた、モード、表示について簡単に説明します。

1．電源のオン／オフ(入／切)と表示の濃度調整

(1) 電源のオン／オフ（入／切）

BREAK
ON（本体右上）を押してください。電源が入って次の表示（画面）になります。

```
RUN MODE                 RUN
>
                         CAPS
                         DEG
BUSY
BATT
```

RUNモード（ランモード：計算やプログラムを実行できる状態）になっていることを示します。
＞マークはプロンプト記号といい、本機の準備が完了したことを示しています。

OFF（ON の左）を押せば電源が切れます。

電源を入れたり切ったりしたときに、一瞬、画面が黒くなったり不要な点や線またはシンボルが点灯することがありますが、機能などには問題ありません。

● オートパワーオフ機能について
電池の消耗を少なくするため、計算実行中（"BUSY"シンボル点灯中）以外の状態で、約11分間新たなキー操作を行わないと、自動的に電源が切れます。ON を押せば電源が入ります。

(2) 表示の濃度調整

表示が見やすくなるように表示濃度（コントラスト）を調整してください。
SHIFT を押したまま ANS（コントラスト）を押します。

表示濃度の調整は、いずれかのモード（18ページ参照）の初期画面で行ってください。

```
*** LCD CONTRAST ***
  ▲   DARK
  ▼   LIGHT
```


▲ を押すと濃くなり、▼ を押すと薄く（淡く）なります。
● 濃度調整が終わったら、BASIC、TEXT などのモードキーを押します。
（次の「2．モードについて」を参照してください。）

2．モードについて

本機には、大きく分けて次の７つのモード（状態）があります。

RUNモード（実行モード）　：ＢＡＳＩＣプログラムを実行したり、マニュアル操作（手操作）での計算などを行います。
PROモード（プログラムモード）　：ＢＡＳＩＣプログラムの入力、編集などを行います。
TEXTモード（テキストエディタモード）　：テキストプログラムの入力、編集、削除、またＳＩＯなどへの出力、読み出し、ＢＡＳＩＣプログラムとテキストプログラムの変換、ソースプログラムの入力などを行います。
ASMBLモード（アセンブラモード）　：機械語のアセンブルを行います。
CASLモード（キャッスルモード）　：ＣＡＳＬのソースプログラムのアセンブル、実行を行います。
PICモード（ピックモード）　：ＰＩＣ用プログラムの作成などを行います。
C言語モード　：Ｃ言語のコンパイルや実行を行います。

モードの切り替えは右上にある緑色の BASIC と TEXT のキーと左端にある SHIFT および A、C、P で行います。
指定されているモードを表すシンボル（ＲＵＮ、ＰＲＯ、ＴＥＸＴ、ＣＡＳＬ）が画面の右側に点灯します。ＡＳＭＢＬモード、Ｃ言語モードおよびＰＩＣモードを表すシンボルは存在しません。

モードの切り替え

設定するモード	キー操作
ＲＵＮモード	BASIC
ＰＲＯモード	BASIC または BASIC BASIC
ＴＥＸＴモード	TEXT
ＡＳＭＢＬモード	SHIFT ＋ ASMBL を押してから A
ＣＡＳＬモード	SHIFT ＋ ASMBL を押してから C
ＰＩＣモード	SHIFT ＋ ASMBL を押してから P
Ｃ言語モード	SHIFT ＋ TEXT（C）

←ＲＵＮモード以外から切り替えるときは BASIC を２回押す。

● たとえば SHIFT ＋ ASMBL は SHIFT を押したままで BASIC（ASMBL）を押すことを示します。なお、この場合 2nd F ASMBL と押しても同じです。
● 統計モードについては62ページを参照ください。
● 機械語モニタモードについては274ページを参照ください。

3．基本的なキー操作

それでは、アルファベットや記号を入力してみましょう。
いったん OFF を押して電源を切ってから、改めて ON を押して電源を入れてください。RUNモードになります。そして、CLS を押して表示内容を消去し、次のようにキーを押してみてください。
キーを押しまちがえたときは、CLS を押して、最初から操作してください。

(1) アルファベット（大文字）の入力

次のキーを押してください。

A B C D E F G

```
ABCDEFG_
```

このように、アルファベットの大文字が表示されます。
ＡＢＣＤＥＦＧと表示されないときは、次の確認をしてください。
①画面右側に“ＣＡＰＳ”シンボルが点灯していますか。点灯していないときは CAPS を押して点灯させてください。
②画面右側に“カナ”シンボルが点灯していませんか。点灯していたら カナ を押して消してください。

(2) アルファベット（小文字）の入力

次にアルファベットの小文字を入力してみましょう。
CAPS を押して画面右側の“ＣＡＰＳ”シンボルを消して、次のキーを押してください。

H I J K L M N

```
ABCDEFGhijklmn_
```

このように、“ＣＡＰＳ”シンボルを消して、アルファベットキーを押せばアルファベットの小文字が表示されます。再び大文字を入力する場合は、“ＣＡＰＳ”シンボルを点灯させてください。

(3) 記号の入力

次に、＃＄％など、キーの上側に橙色で書かれている記号を入力してみましょう。
これら、橙色で書かれている記号を入力するときは、次の２つの方法があります。
① SHIFT を押したまま、これらの記号が書かれているキーを押す。
②これらの記号が書かれているキーを押す前に 2nd F を押す。
まず、SHIFT を押したまま、次のキーを押してください。

E(#) R($) T(%)

```
ABCDEFGhijklmn#$%_
```


次に、各キーを押す前に 2nd F を押してください。

2nd F I(<) 2nd F O(>) 2nd F P(@)

```
ABCDEFGhijklmn#$%<>@_
```

（2nd F を押したとき、画面右側に“2nd F”シンボルが点灯します。）
このように橙色で書かれている記号を入れるとき、または橙色で書かれている機能を使うときは、SHIFT を押したまま、それぞれのキーを押すか、それぞれのキーを押す前に、2nd F を押します。

次に数字を入れてみましょう。

1 2 3 4 5 6
7 8 9

```
ABCDEFGhijklmn#$%<>@1234
56789_
```

● １行24桁を超えると、次の行に入ります。

メモ　入力について

キーを押して、数値や文字を計算機に記憶させることを「入力」や「置数」などといいます。

(4) カーソルについて

次に、◀ を１回押してください。すると“9”のところで ■ マークが点滅します。

```
ABCDEFGhijklmn#$%<>@1234
56789
```

次に ◀ を押したままにしてみます。
しばらくすると、■ マークが左に移動していきます。左端まで行くと上の行の右端へ移り、再び左へ移動します。１行目の左端で止まります。
次に ▶ を押したままにします。■ マークが右に移動していき、最後に ■ マークが消えて＿マークが点灯します。

この ■ マークと＿マークは、カーソルといわれるもので、キーを押したときに文字などが入る位置を示しています。
キーを押しまちがえて、違う文字や記号が入ってしまったときは、カーソルをまちがった文字などの位置に移動させて、正しいキーを押せば訂正できます。（くわしくは26ページを参照してください。）

こんどは ▲ を押してみてください。カーソルが１行上（１行目）へ移動します。
次に ▼ を押してみてください。カーソルが１行下（２行目）へ移動します。
このように、▲ ▼ はカーソルを上下に動かします。

カーソルが先頭行にあるときに[▲]を押すと、その行の先頭（左端）へ移動します。
カーソルが最後の行にあるときに[▼]を押すと、その行の最後へ移動します。

キーの記載方法について

● この取扱説明書では、キー操作またはキーの機能を次のように記載します。

[BASIC]（上側にASMBL）の例

①キーに書かれている機能を使用する場合は、単に[BASIC]と記載します。

②キーの上側に橙色で書かれている機能を使用する場合は、[SHIFT]＋[ASMBL]または[2nd F][ASMBL]と記載します。

なお、[SHIFT]＋[ASMBL]は[SHIFT]を押したまま[ASMBL]を押すことを意味します。

また、[2nd F][ASMBL]は[2nd F]を押して離した後、[ASMBL]を押すことを意味します。

● ただし、数字やアルファベット、記号、カタカナなど、単にキーや本体（キーの上側）に書かれている文字や数字、記号を入力するだけの場合は、キーの枠や[SHIFT]、[2nd F]の操作などを省いて記載します。

〈例〉 [S][H][A][R][P] → SHARP
[A][S][SHIFT]＋[$] → AS$

● スペースキーは[SPACE]と記載します。

● この計算機は、数字の0をØと表示します。
これは、アルファベットのオー（O）と区別するためです。本書でもオーとゼロの区別がつきにくい場合はゼロをØと表しています。

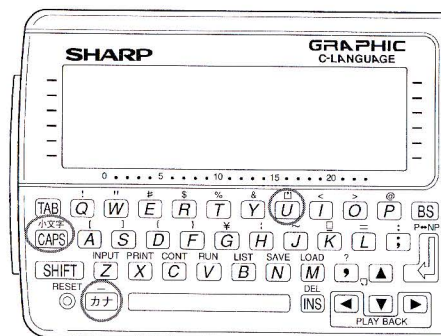
なお、説明上必要な場合には、[CLS]（上側にCA）などの方法で示しています。

4．表示シンボルについて

本機には色々な機能やモードがあります。そして、いま本機がどのような状態になっているのかが分かるように、画面の左や右にシンボルを点灯させます。
次に表示シンボルとその意味を示します。
なお、本書の表示例では、説明上必要なシンボルだけを記載しています。

＜ シンボル ＞	＜ 説　　明 ＞
BUSY	：計算実行中または命令実行中です。
BATT	：電池が消耗していることを示します。このシンボルが点灯したときは、速やかに電池を交換してください。
RUN	：BASICのRUNモード（実行モード）になっていることを示します。 [BASIC]で点灯できます。
PRO	：BASICのPROモード（プログラムモード）になっていることを示します。 [BASIC]で点灯できます。
TEXT	：TEXTモード（テキストモード）になっていることを示します。 [TEXT]で点灯できます。
CASL	：CASLモード（キャッスルモード）になっていることを示します。 [2nd F][ASMBL]（または[SHIFT]＋[ASMBL]）と押してから[C]を押せば点灯できます。
STAT	：統計モードになっていることを示します。 [2nd F][STAT]（または[SHIFT]＋[STAT]）と押せば、点灯／消灯ができます。
2nd F	：[2nd F]が押されたことを示します。 このシンボルが点灯しているとき、各キーの橙色で書かれている機能が指定されていることを示します。
M	：マニュアル（手操作）による計算用のメモリに0以外の数値が入っていることを示します。
CAPS	：アルファベットの大文字が入力できる状態になっています。このシンボルが消えているときは、アルファベットの小文字が入力できる状態になっています。この状態は[CAPS]で切り替えます。 ただし、“カナ”シンボルが点灯しているときは、カナの入力が優先され、アルファベットは入力されません。
カナ	：ローマ字のつづりでキーを押せば、カタカナを入力できます。（23ページと382ページ参照） [カナ]を押すことにより、このシンボルの点灯／消灯ができます。
小	：小さいカナ文字（ァィゥェォャュョッ）を入力することができます。 “カナ”シンボル点灯時、[CAPS]でこのシンボルを点灯できます。
DEG RAD GRAD	：三角関数、逆三角関数、座標変換を行うときに使用する角度の単位を示します。（関数計算の項を参照）
CONST	：定数が記憶されていることを示します。（定数計算が設定されていることを示します。36ページ参照） このシンボルが点灯しているときは、[↵]で定数計算が行われます。 不要なときは[2nd F][CA]（[SHIFT]＋[CA]）と押して、定数を消去してください。
PRINT	：プリント（印字）モードになっていることを示します。 プリンタが接続されているときに、[2nd F][P↔NP]と押せば、点灯／消灯ができます。

5．カタカナの入力のしかた

本機にはローマ字のつづりで入力した文字をカナに変換する“ローマ字→カナ変換機能”があります。カナは左下にある［カナ］を押して、画面右側に“カナ”シンボルを点灯させ、アルファベットキーを用いて入力します。

［カナ］……………このキーで“カナ”シンボルの点灯／消灯を行います。“カナ”が点灯しているときに、カナを入力できます。

［小文字 CAPS］……………ァィゥェォャュョッ（小さいカナ文字）を入力するときに使います。

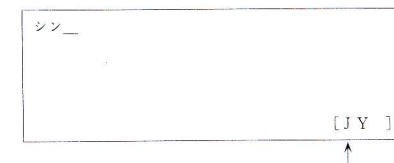
［SHIFT］＋［U］……「ン」を入力するときに使用します。

＜例＞	（入力文字）		（キー操作）
	カタカナ	→	KATAKANA
	ガッコウ	→	GAKKOU
	ヘンカン	→	HENKAN［SHIFT］＋［U］
	ハンイ	→	HAN［SHIFT］＋［U］I
	ディスク	→	DHISUKU
	ウォッチ	→	U［小文字 CAPS］OTTI

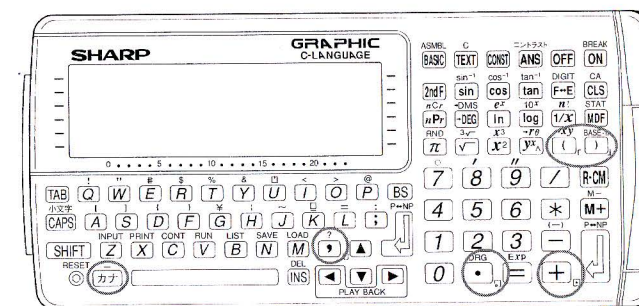
● 「ン」を入力するときに、［N］の後にYを除く子音がくる場合は［SHIFT］＋［U］を押す必要はありません。

● ローマ字の表記の方法については、382ページを参照してください。

● カナ入力中は、押したキー（子音）が画面右に表示されます。

＜例＞SINJY

ジュを入力するためにJYと押したところです。Uを押すとカーソル位置にジュが表示され、この表示は消えます。この表示は1行目に出る場合もあります。

カナ記号の入力のしかた

カナ記号は次のキー操作により入力します。

ー（長音符）	：［SHIFT］＋［カナ］
、（読点）	：［，］
。（句点）	：［・］
・（中点）	：［＋］
「（カギカッコ）	：［（］
」（カギカッコ）	：［）］

第１章
マニュアル計算（手操作による計算）

1．マニュアル計算

マニュアル計算とは、１つ１つの計算を１回１回手操作によって計算することをいいます。関数電卓として使用する場合とほぼ同じですが、ＢＡＳＩＣプログラムを組む前に、必ず知っておかなければならないことについて説明します。60進数の度・分・秒計算機能は、マニュアル計算でのみ可能です。
①計算式の作りかた
②まちがえたときの訂正のしかた
③連続計算のしかた
④関数の使いかた
など、基本的で重要な事項です。よく読んで理解し、本機の操作に慣れ、いろいろな基本計算をマスターしてください。

2．マニュアル計算のしかた

(1)電源オン

[ON] を押して、電源を入れます。

“ＲＵＮ”シンボル*が点灯します。これはマニュアル計算ができることを示しています。
もし、ＲＵＮのシンボルが点灯していないときは [BASIC] を押してＲＵＮを点灯させてください。
※説明上必要な場合を除き、「シンボル」の文字は省略します。

■計算の記号について

通常の数学では四則計算の記号として「＋、－、×、÷」を使いますが、ＢＡＳＩＣ言語では「×」や「÷」の代りに「＊」や「／」などの記号を用います。また、本機ではべき乗や指数、その他の関数などにおいて、通常の数学で用いられている記号とは違うかたちで表現されていますので、以下に本機のマニュアル計算やＢＡＳＩＣ言語で使用する計算の記号や特殊記号について説明します。

	計　算	数学で用いる記号	本機で用いる記号	読　み　か　た
(1)	加　算	＋	＋	プラス
(2)	減　算	－	－	マイナス
(3)	乗　算	×	＊	アスタリスク
(4)	除　算	÷	／	スラッシュ
(5)	整数の除算		¥	エン
(6)	整数の剰余		MOD	モッド
(7)	符　号	－または＋	－、＋	マイナス、プラス
(8)	計算の実行	＝	[↵]	キャリッジリターン

〔補足〕割る数と割られる数の両方の、小数点以下第１位を四捨五入して、整数化してから除算を行います。¥では商が求まり、ＭＯＤでは余りが求まります。

〈例〉
51¥5 → 10　　51 MOD 5 → 1　　(51÷5＝10…1)
51¥－5.7 →－8　　51 MOD －5.7 → 3　　(51÷－6＝－8…3)
87.57¥5.4 → 17　　87.57 MOD 5.4 → 3　　(88÷5＝17…3)
30°36′¥14°36′ → 2　　30°36′ MOD 14°36′ → 1　　(31÷15＝2…1)
30°36′¥4.4 → 7　　30°36′ MOD 4.4 → 3　　(31÷4＝7…3)
87.57¥14°36′ → 5　　87.57 MOD 14°36′ → 13　　(88÷15＝5…13)

3．キー操作の練習（訂正のしかた）

ここで簡単な例題をもとに、キー操作の練習をします。
数値などを入れているときに違った数字や命令を入れたり、余分な数字や命令を入れたり、あるいは抜かしてしまったときの訂正のしかたです。

〔例題〕①

計算式	キー操作	答
5＋15×2÷4	5 [＋] 1 5 [＊] 2 [／] 4 [↵]	12.5

(1)まちがって他のキーを押したとき

5＋15×2÷4　を　5＋15×2×4　とした場合

キー操作	表示部
5 [＋] 1 5 [＊] 2 [＊] 4	5＋15＊2＊4_ ← ここがちがう
[◀][◀]（２回押す）このキーを押して左へカーソル*を移動させる	5＋15＊2＊4 ← この部分にカーソルを移動させて点滅させる
[／]（この後 [↵] としてもよい）	5＋15＊2／4 ← 正しいキーを入れる　ここが点滅する

[▶]　このキーを押して右へカーソルを移動させる	5＋15＊2／4_
[↵]	12.5 答
キー入力した式を確認したいとき（これをプレイバック機能という）	
[◀]　このキーを押すと	5＋15＊2／4_
[↵]　を押して	12.5
[▶]　さらにこのキーを押すと	5＋15＊2／4 ↑この部分が点滅する
[↵]	12.5

※カーソルについて

キー操作を行うとき表示部に ＿ が表示されたり、キー操作の訂正のときに ▒ マークが点滅したりしますが、これらのマークはカーソルと呼ばれるもので、次に操作したキーがこの部分に入ることを示しています。つぎに説明する挿入モードでの ▫ もカーソルです。

■まちがって余分なキーを押したとき

〔例題〕②

計算式	キー操作	答
20×3÷5	20[＊]3[／]5[↵]	12

キー操作	表示部
20[＊]3[＊][／]5	2Ø＊3＊／5_ ↑余分なキー ↑数字の0（ゼロ）と英文字のO（オー）とを区別するために　0（ゼロ）→ Ø と表示する
[◀][◀][◀]このキーを3回押す	2Ø＊3＊／5 ↑この部分にカーソルを移動させる
[SHIFT]＋[DEL] （下記の※を参照）	2Ø＊3／5 ↑＊が削除されてここが点滅する
[↵]	12.
キー入力した式を確認したいとき [◀]	2Ø＊3／5_

※[SHIFT]＋[DEL]は[SHIFT]を押したまま[INS](DEL)を押すことを示します。なお、[2nd F][INS](DEL)と押してもDEL機能（削除）が働きます。

(3)プレイバック機能について

〔例題〕②において、余分なキーを押したのに気がつかずに[↵]を押してしまうと、エラーが発生します。このようなとき、キー入力した式を呼び戻して訂正する機能がプレイバック機能です。この機能を使うとエラーを起こした場所をカーソルで示しますので、エラーの原因がわかりやすく大変便利です。

キー操作	表示部
20[＊]3[＊][／]5[↵]	ERROR　10
[◀]	2Ø＊3＊／5 ↑ここが点滅する 例題の式は20×3÷5なので この場合＊が不要です
[BS] （この後[↵]としてもよい）	2Ø＊3／5 ↑カーソルの前の＊が削除されます
[▶][▶]　2回押して	2Ø＊3／5_ ↑カーソルが最後に表示されます
[↵]	12.

(4)キー操作が抜けたとき

〔例題〕③

計算式	キー操作	答
2×3÷10	2[＊]3[／]10[↵]	0.6

キー操作	表示部
23[／]10	23／1Ø_ ↑ここに＊が抜けている
[◀][◀][◀][◀] カーソルを左へ4つ移動	23／1Ø ↑この部分を点滅させます
[INS] 訂正モードから挿入モードに変わります	23／1Ø ↑カーソルが ▒ から ▫ に変わり、次に入力した文字がカーソルの前に挿入されます
[＊] （この後[↵]としてもよい）	2＊3／1Ø ↑＊が挿入（追加）されます
[▶][▶][▶][▶] カーソルを右へ4つ移動	2＊3／1Ø_
[↵]	Ø.6

(5)数式において誤りが多いとき

[CLS]を押して、今まで入れた内容をすべて消し、あらためて最初から入力し直します。

(6)DELとINS、BSの意味

DEL　→　DELETE……（デリート）削除を意味します。

INS　→　INSERT……（インサート）挿入を意味します。

入力した文字がカーソルの前に挿入されます。再度[INS]を押すと訂正

モードに変わります。また、[↵]を押したときも挿入モードは解除され訂正モードに戻ります。

BS　→　BACKSPACE……（バックスペース）１文字分の後退を意味します。カーソルの前の文字が削除されます。

練習問題	1

を行ってください。

練習問題は74ページから記載しています。

4．エラーの処理について

[練習問題 1]を行って、すぐに答えの出た人もいるでしょう。また、答えが出ずにＥＲＲＯＲ１０などという表示が出た人もいることでしょう。

ＥＲＲＯＲ（エラー）が出た人も出なかった人も、次の例題でエラーの処理を行ってみてください。

〔例題〕④

$$\frac{36}{2+4}$$

キー操作	表示部
３６[／]２[＋]４[)]	３６／２＋４）_
[↵]	ＥＲＲＯＲ　１０
[◀]	３６／２＋４） ↑ここが点滅します この意味は[)]（カッコ）の部分がエラーの原因であることを示しています
この〔例題〕④の場合は36／（２＋４）が正しい式ですから	
[◀][◀][◀]	３６／２＋４） ↑ここが点滅します
[INS]　挿入モードにします	３６／２＋４）
[(]	３６／（２＋４）
[↵]	６．
式を確認したいときは [◀]	３６／（２＋４）_ ↑カーソルが最後に表示されます
または [▶]	３６／（２＋４） ↑カーソルが先頭に表示されます

メモ　（　）について

分数においては、式にカッコがついていなくても、分母、分子はそれぞれ全体をカッコでくくる必要があります。ただし、分母、分子とも数値が１つだけの場合は省略できます。

なお、関数キーを使う場合（45ページ参照）も中カッコ｛　｝等を必要とする場合があります。

計算機では小カッコも中カッコも同じ[(][)]のキーを使います。

5．数式が長い場合の連続計算のしかた

〔例題〕⑤

キー操作	表示部
２[＊]３[／]１０[↵] →これで区切り計算となります	０．６
[＊]１００	０．６＊１００_
[↵]	６０．

計算式の途中で[↵]を押すと、それまでの数値計算が実行されるので、区切りのよいところで[↵]を押します。続いて計算記号と数値をキー入力すれば、長い数値の計算式でも計算が容易になります。式が表示上で255字分以内であれば、区切ることなく計算できます。

練習問題	2

を行ってください。

6．ラストアンサー機能について

これは、計算した結果を記憶している機能です。具体的にはマニュアル計算で[↵]を押して得られた結果や、ダイレクトアンサー機能を用いて得られた結果（これらをラストアンサーと呼ぶ）などを記憶します。

この機能は、次のような使いかたをすると便利です。

ダイレクトアンサー機能：[↵]を使用せずに関数計算を行う方法。
数値を入れてから、各関数キーを押す。（38ページ参照）

〔例題〕⑥

$\dfrac{4\times6}{4+6}=2.4$ この2.4という結果を使って

$\dfrac{1}{2.4}+\dfrac{3}{2.4}+\dfrac{2}{2.4}$ のような計算を行う場合

キー操作	表示部
4 [*] 6 [/] [(] 4 [+] 6 [)] [↵]	2.4
1 [/] [ANS] [+] 3 [/] [ANS] [+]	1/2.4+3/2.4+_
2 [/] [ANS]	1/2.4+3/2.4+2/2.4_
[↵]	2.5
ここで [↵] を入れたことによって、新しい計算を実行したことになりますのでラストアンサーは、2.5という数値に入れ替わります。	

ラストアンサー機能とは、ひとつの記憶箱（レジスタ）にひとつのデータをしまっておくことであり、常に一番新しいデータだけが記憶できます。

なお、ラストアンサーはPROおよびRUNモードのとき、[ANS] で呼び出すことができます。

7．計算結果の表示方法

いろいろな計算を実行する中で、小数部については小数点以下2桁、あるいは3桁くらいで十分という場合が多いものです。また、小数点の表示ではなく、指数での表示のほうが便利でわかりやすい場合もあります。

そこで、計算結果の表示方法について、本機の便利な利用法を説明します。

通常用いている数値は、

12300や0.0987のように表されます。

しかし、理科や工科系においては、

12300を　1.23×10^4

0.0987を　9.87×10^{-2}

と、表すことが多くあります。

数値をこのように表す方法を**指数方式**と呼びます。

1.23×10^4

9.87×10^{-2}

この部分を**指数部**といいます。（10^{-2} の部分）

この部分を**仮数部**といいます。（9.87 の部分）

本機では、数値を指数方式で表示する場合は、指数部を示す記号としてE（[2nd F][Exp] で入力）を用います。次の例題でキー操作の練習をしてください。

キー操作	表示部
例題A $1.2\times10^9=$	
1 [・] 2 [2nd F] [Exp] 9	1.2E9_
[↵]	1200000000.
[F↔E]	1.2E 09
普通の表示を指数表示にします	
例題B $1.2\times10^{10}=$	
1 [・] 2 [2nd F] [Exp] 1 0 [↵]	1.2E 10
[F↔E]	1.2E 10
	ここが2桁になると [F↔E] を押しても変わりません
例題C 0.0987=	
0 [・] 0 9 8 7 [↵]	0.0987
[F↔E]	9.87E−02
	10^{-2} を意味します
例題D $\dfrac{1.71\times2.43\times10^3}{3.25\times1.5\times10^{-2}}=$	
1 [・] 71 [*] 2 [・] 43 [2nd F] [Exp] 3 [/] [(] 3 [・] 25 [*] 1 [・] 5 [2nd F] [Exp] − 2 [)] [↵]	85236.92308
[F↔E]	8.523692308E 04

（注）$a\times10^x$ をa [2nd F] [Exp] x と置く場合、x は±(Ø～99)の整数でなければなりません。整数3桁以上の数値を入れたときは、下から2桁をxと読んでしまいます。また、小数部分をもった数値を入れたときは、ERROR10になります。

〈例〉 1 [2nd F] [Exp] −1125 [↵] → 1. E−25（誤り）

53ページ「[2nd F] [10^x]、[2nd F] [e^x]、[2nd F] [Exp] の違いについて」を参照。

メモ　大きな数字、小さな数字について

この計算機は10桁の計算機です。値の大きな数字（指数部が10以上）や小さな数字（指数部が−10以下）は通常の表示方法では表示できません。そのため、[F↔E] を押しても表示は変わらず、常に指数方式で表示されます。

8．表示に関するフォーマット指定

(1)DIGIT指定

マニュアル計算で得られた結果などでは、特殊計算を除いて小数部は2桁や3桁などでよい場合が多いものです。そこで、計算する前あるいは計算結果が表示されているときにDIGIT（デジット）によって計算結果の小数部の桁数を指定することができます。

〈例〉 [2nd F] [DIGIT]　0　整数を表示　(小数点以下１桁目を四捨五入)
[2nd F] [DIGIT]　2　小数点以下２桁まで表示 (小数点以下３桁目を四捨五入)
[2nd F] [DIGIT]　4　小数点以下４桁まで表示 (小数点以下５桁目を四捨五入)
[2nd F] [DIGIT]　[・]　デジット指定解除　電源オフやモードの切り替えなどでも解除

(注) ● 結果が60進数の度・分・秒の記号で表示されたときは、DIGIT指定は無効です。

〔例題〕⑦

$\dfrac{1.71}{3.5}$ の計算を小数点以下２桁まで求めなさい。

キ　ー　操　作	表　　示　　部
まず、デジット指定で小数部の桁数を指定します。 [2nd F] [DIGIT] 2 1 [・] 71 [／] 3 [・] 5 [↵]	 0.49
1.71／3.5の結果は0.488571428ですが小数部分を２桁に指定していますので、３桁目が四捨五入されて0.49となります。	

■ＵＳＩＮＧ命令

ＵＳＩＮＧ（ユージング）命令でフォーマットの指定をするときは、計算の前にあらかじめ指定しておく必要があります。最後の桁は切り捨てられた値となります。

①ＵＳＩＮＧ　"###"　符号と整数２桁を表示
②ＵＳＩＮＧ　"###. "　符号と整数２桁と小数点を表示
③ＵＳＩＮＧ　"###. ##"　符号と整数２桁と小数点と小数点以下２桁を表示
④ＵＳＩＮＧ　"###, ###."　符号、３桁区切りマーク（,)、整数５桁と小数点を表示
３桁区切りマークも１桁と数えます。たとえば－1,234,567. を表示させるときは、ＵＳＩＮＧ　"######, ###."のように#と, をあわせて最低10個指定する必要があります。
⑤ＵＳＩＮＧ　"##. ##∧"　小数点以下２桁までの指数方式で表示
このとき指数部はＥと符号を含めて４桁（Ｅ－００）が、自動的に取られます。
⑥ＵＳＩＮＧ　フォーマット指定を解除

(注) ● 符号が正のときはスペースになります。
● ＵＳＩＮＧ命令を実行すると、デジット指定は解除されます。
同様にデジット指定により、ＵＳＩＮＧの指定は解除されます。
● 結果が60進数の度・分・秒の記号で表示されたときは、ＵＳＩＮＧ命令は無視されます。

〔例題〕⑦をＵＳＩＮＧを使用して計算してみましょう。

キ　ー　操　作	表　　示　　部
[U] [S] [I] [N] [G] [SHIFT] + ["] [SHIFT] + [#]	USING "#_
[SHIFT] + [#] [・] [SHIFT] + [#]	USING "##. #_
[SHIFT] + [#] [SHIFT] + ["]	USING "##. ##"_
[↵]	>
1 [・] 7 1 [／] 3 [・] 5 [↵]	0.48 これは「+0.48」ということを意味しています
またここで [ANS] を押すと	4.885714286E－01_ ＵＳＩＮＧ指定でない表示がでます
そしてさらに [↵] を押すと	0.48 ＵＳＩＮＧ指定に戻ることができます

● 計算が終わったら、[U] [S] [I] [N] [G] [↵] と押してＵＳＩＮＧ指定を解除してください。

練習問題　３　を行ってください。

９．数値丸め機能　ＭＤＦ（モディファイ）

次の計算をしてみましょう。

$\dfrac{55.4}{9} \times 9$

キ　ー　操　作	表　　示　　部
([2nd F] [CA])	
55.4 [／] 9 [↵]	6.155555556
[*] 9	6.155555556*9_
[↵]	55.4

この計算例では小数部がたくさん表示されるのですっきりせず、わずらわしく感じます。そこで有効桁数を考え、式の途中の結果の端数を四捨五入し、しかも、最終結果においてより確からしい値を求める方法があります。これを数値丸め機能（ＭＤＦ）といいます。

先の例において
55.4÷９＝6.155555556ですが、小数第４位を四捨五入し6.156として
6.156×９を計算すれば55.404となります。

キ　ー　操　作	表　　示　　部
[2nd F] [DIGIT] 3	
55.4 [／] 9 [↵]	6.156
[MDF]	6.156
[*] 9	6.156*9_
[↵]	55.404

このように数値丸め機能を用いれば、計算途中で有効桁数の端数処理を行いながらより確からしい値を求めることができます。
ただし、この機能（ＭＤＦ）はＤＩＧＩＴまたはＵＳＩＮＧの指定がされているときだけ有効です。
（注）● 結果が60進数の度・分・秒の記号で表示されたときは、この機能は無効です。

10. 計算結果の符号の反転

計算によって得られた結果を次の計算に利用する場合、その結果の符号を反転させてから使用したいときに便利な機能です。
たとえば、1572円、2350円、3058円の買い物をして10000円で支払う場合、買い物の合計金額とお釣りの金額を求めるときの操作は次のようになります。

キー操作	表示部
1572 [+] 2350 [+] 3058 [↵]	6980.
[2nd F] [(−)]	−6980.
[+] 10000 [↵]	3020.
（注）A－B＝－B＋Aにより10000－6980を－6980＋10000として計算	
または	
1572 [+] 2350 [+] 3058 [↵]	6980.
[−] 10000 [↵]	−3020.
[2nd F] [(−)]	3020.

11. メモリ計算

マニュアル計算で、数値を一時記憶したり、計算結果を累計できるメモリがあります。
このメモリは次のキー操作で利用できます。

[R･CM] ……………メモリの内容を表示します。
続いてもう１回押せばメモリの内容を消去します。メモリ計算は、このキー操作でメモリ内を消去してから始めます。

[M+] ………………表示されている数値をメモリに加えます。また、式が表示されているときは、その式を計算して、結果をメモリに加えます。

[2nd F] [M−] ……表示されている数値をメモリから減算します。また、式が表示されているときは、その式を計算して、結果をメモリから減算します。

メモリについて

メモリとは数値（お金）をたくわえておく金庫のようなものです。金庫に次々と数値を追加したり（[M+]）、必要な分だけ引き出したり（[2nd F] [M−]）することができます。また、金庫にたくわえられている値（全額）を呼び出す（[R･CM]）こともできます。
R･CMとはRM（Recall Memory メモリの呼び出し）とCM（Clear Memory メモリの消去）のことです。この２つの働きを１つのキーにまとめて表しています。

〔例題〕⑧

(1)　24×34、64÷５、13×28÷８の計算を行い、その結果の合計を求めなさい。
(2)　次の計算を行いなさい。

　　24＋46＝
－）37×４＝
＋）84－29＝
　　合計

キー操作	表示部	備考
[R･CM] [R･CM] [CLS]		メモリ内容を消去
24 [*] 34 [M+]	816.	計算結果をメモリに加えます。
64 [/] 5 [M+]	12.8	このとき、メモリが使われていることを示す [M] シンボルが点灯します。
13 [*] 28 [/] 8 [M+]	45.5	
[R･CM]	874.3_	合計が求まります。
[R･CM] [R･CM] [CLS]		メモリ内容を消去（[M] シンボル消灯）
24 [+] 46 [M+]	70.	
37 [*] 4 [2nd F] [M−]	148.	37×４の計算を実行し、その結果をメモリから減算します。（※参照）
84 [−] 29 [M+]	55.	
[R･CM]	−23._	合計を呼び出します。

※次のように操作することもできます。
　37 [*] 4 [↵]　　37×４を計算し、結果を表示
　[2nd F] [M−]　　表示されている数値をメモリから減算

12. 定数計算機能

定数計算とは、ある決まった値（定数）に対して決まった計算を行うことです。次のキーにより定数と計算を設定できます。

[CONST] ………定数と計算を設定

(1)設定方法　　　（注）ａは定数とする数値（または式）を示します。

加算：[+] ａ [CONST]（＋ａが定数）　またはａ [+] [CONST]（ａ＋が定数）
減算：[−] ａ [CONST]（－ａが定数）　またはａ [−] [CONST]（ａ－が定数）
乗算：[*] ａ [CONST]（×ａが定数）　またはａ [*] [CONST]（ａ×が定数）
除算：[/] ａ [CONST]（÷ａが定数）　またはａ [/] [CONST]（ａ÷が定数）

定数計算を設定すると表示部右側に“ＣＯＮＳＴ”が点灯します。

(2)設定内容の確認

“ＣＯＮＳＴ”が点灯しているときに次のキーを押せば四則の記号（＋－＊／）も含めて設定内容が表示されます。

[2nd F] [CONST]（[SHIFT] ＋ [CONST]）

(3)設定内容の消去

次のキー操作で設定内容が消去されます。また、電源を切ったときも消去されます。

[2nd F][CA]（[SHIFT]＋[CA]）………“ＣＯＮＳＴ”が消灯します。

〔例題〕⑨

(1)　＋(4.8＋3.6)を定数とする、次の計算を行いなさい。 32.5＋(4.8＋3.6)＝ 24－18.5＋(4.8＋3.6)＝ 8.2×６＋(4.8＋3.6)＝ (2)　(57－14)×を定数とする、次の計算を行いなさい。 (57－14)×18＝ (57－14)×(27／４)＝ (57－14)×(24＋43)＝

キ　ー　操　作	表　示　部	備　　　考
[＋][（]4.8[＋]3.6[）]	＋（４．８＋３．６）＿	
[CONST]	＞	定数計算設定（“ＣＯＮＳＴ”点灯）
32.5[↵]	４０．９	
24[－]18.5[↵]	１３．９	
8.2[＊]６[↵]	５７．６	
[2nd F][CONST]	＋８．４＿	設定内容確認（4.8＋3.6の結果が定数になっています）
[（]57[－]14[）][＊]	（５７－１４）＊＿	
[CONST]	＞	定数計算設定（前の設定内容は消されます）
18[↵]	７７４．	
27[／]４[↵]	２９０．２５	
24[＋]43[↵]	２８８１．	
[2nd F][CA]	＞	設定内容消去

第２章
関数計算の操作方法と練習

1．マニュアル（手操作）での関数計算

マニュアルにおいて、関数計算を単独で行う場合の方法をここで説明します。
本機には、ＢＡＳＩＣ言語でも使える基本関数が多くありますが、それらについてはＢＡＳＩＣプログラムの項で説明します。
マニュアル計算はＲＵＮモードで行います。[BASIC]で画面の右上に“ＲＵＮ”を点灯させてください。
ＰＲＯモードでマニュアル計算はできません。
60進数の度・分・秒計算機能は、マニュアル計算でのみ可能です。

関数キーを使う方法

マニュアル計算の場合は、通常、関数キーを使います。

ＲＵＮモード ①　**関数キー**を押して、続いて**数値**を入れ[↵]を押す。 ②　**数値**を入れ、続いて**関数キー**を押す。 ②の場合、関数電卓と同じような使いかたとなり、[↵]を押さなくても計算結果が出ます。これをダイレクトアンサー機能といいます。ただし、この機能は、べき乗や座標変換のように２つの数値が必要な関数、また式の入力途中では働きません。 表の中にこのダイレクトアンサー機能を（DAns 機能）として示しています。

アルファベットキーを使う方法

関数キーを用いなくても、アルファベットで表現されるものは、アルファベットキーで入力することもできます。関数キーが用意されていない関数はアルファベットキーで入力します。
たとえば、
5^2 の計算では、[x^2] ５と操作しても、[S][Q][U] ５と操作しても答は同じです。プログラムの中で関数を扱う場合は、アルファベットの使用頻度が高いため、この方法が便利なときがあります。

(1)関数入力の方法 1

（注）DAns：ダイレクトアンサー

項目	通常の表現	本機の表現	読みかた	入力方法と備考
2乗	x^2	SQU	スクウェア square （平方）	[x^2] X [↵] または X [x^2]（DAns 機能）
平方根	$\sqrt{x}$	SQR	ルート root square root （平方根）	[√] X [↵] または X [√]（DAns 機能）
3乗	x^3	CUB	キュービック cubic （立方）	[2nd F] [x^3] X [↵] または X [2nd F] [x^3]（DAns 機能）
立方根	$\sqrt[3]{x}$	CUR	キュービックルート cubic root （立方根）	[2nd F] [∛] X [↵] または X [2nd F] [∛]（DAns 機能）
逆数	$\frac{1}{x}$	RCP	レシプロカル reciprocal （逆数）	[1/x] X [↵] または X [1/x]（DAns 機能）
円周率	π	PI	パイ Pi （ギリシャ語・円周率）	[π] [↵] または [P] [I] [↵] $\pi \fallingdotseq 3.141592654$
べき乗 （べき乗根）	y^x （注） $\sqrt[x]{y} = y^{\frac{1}{x}}$	∧	ハットマーク power （累乗，べき乗）	Y [y^x∧] X [↵] （注）Y [y^x∧] [(] 1 [/] X [)] [↵]
絶対値	$\|x\|$	ABS	アブソリュート absolute （絶対値）	[A] [B] [S] X [↵] Xの絶対値
整数化		INT	インテジャー integer （整　数）	[I] [N] [T] X [↵] X以下でかつ最も大きい整数
符号関数		SGN	シグナム sign （符　号）	[S] [G] [N] X [↵] Xの値が正のとき１、負のとき－１、 ０のとき０が得られる

（注）・「入力方法と備考」欄のX、Y、n、rは「通常の表現」欄のx、y、n、rに対応する数値を表します。また、θは角度を表す数値です。
・Xについては、60進数の度・分・秒の記号で指定することもできます。

(2)関数入力の方法 2

項目	通常の表現	本機の表現	読みかた	入力方法と備考
乱数		RND	ランダム random numbers （乱　数）	[2nd F] [RND] X [↵] 乱数を発生する ● Xについては114ページを参照ください。
数値丸め機能		MDF	モディファイ modify （修正する）	[MDF] X [↵] DIGIT またはUSING 命令で指定された桁数の端数処理を行う。
桁数指定		DIGIT	デジット digit	[2nd F] [DIGIT] n nは指定する小数点以下の桁数 $0 \leqq n \leqq 9$ 解除は [2nd F] [DIGIT] [・] と操作
階乗	n!	FACT	ファクトリアル factorial （階　乗）	[2nd F] [n!] n [↵] または n [2nd F] [n!]（DAns 機能）
順列	$_nP_r=\frac{n!}{(n-r)!}$	NPR	パーミュテーション permutation （順　列）	[$_nP_r$] [(] n [,] r [)] [↵] ただし、$n \geqq r$
組み合わせ	$_nC_r=\frac{n!}{r!(n-r)!}$	NCR	コンビネーション combination （組み合わせ）	[2nd F] [$_nC_r$] [(] n [,] r [)] [↵] ただし、$n \geqq r$
対数関数	$\ln x$ または $\log_e x$	LN	ローン natural logarithm （自然対数）	[ln] X [↵] または X [ln]（DAns 機能）
対数関数	$\log x$	LOG	ロガリズム common logarithm （常用対数）	[log] X [↵] または X [log]（DAns 機能）
指数関数	e^x （自然指数）	EXP	エクスポーネント exponent （指数）	[2nd F] [e^x] X [↵] または X [2nd F] [e^x]（DAns 機能） $e \fallingdotseq 2.718281828$
指数関数	10^x （常用指数）	TEN	テン power of ten	[2nd F] [10^x] X [↵] または X [2nd F] [10^x]（DAns 機能）

(3)関数入力の方法 [3]

項目	通常の表現	本機の表現	読みかた	入力方法と備考
三角関数	$\sin\theta$	SIN	サイン sine （正弦）	● [2nd F] [DRG] でDEG、RAD、GRADのいずれかを指定 [sin] θ [↵] または θ [sin]（DAns 機能）
	$\cos\theta$	COS	コサイン cosine（余弦）	[cos] θ [↵] または θ [cos]（DAns 機能）
	$\tan\theta$	TAN	タンジェント tangent（正接）	[tan] θ [↵] または θ [tan]（DAns 機能）
逆三角関数	$\sin^{-1}x$	ASN	アークサイン arc sine （逆正弦）	● [2nd F] [DRG] でDEG、RAD、GRADのいずれかを指定 [2nd F] [$\sin^{-1}$] X [↵] または X [2nd F] [$\sin^{-1}$]（DAns 機能）
	$\cos^{-1}x$	ACS	アークコサイン arc cosine （逆余弦）	[2nd F] [$\cos^{-1}$] X [↵] または X [2nd F] [$\cos^{-1}$]（DAns 機能）
	$\tan^{-1}x$	ATN	アークタンジェント arc tangent （逆正接）	[2nd F] [$\tan^{-1}$] X [↵] または X [2nd F] [$\tan^{-1}$]（DAns 機能）

（注）θ、Xについては、60進数の度・分・秒の記号で指定することもできます。

項目	通常の表現	本機の表現	読みかた	入力方法と備考	
双曲線関数	$\sinh x$	HSN	ハイパボリック サイン hyperbolic sine	[H] [S] [N] X [↵] $\frac{e^{x}-e^{-x}}{2}$	詳細については専門書参照のこと
	$\cosh x$	HCS	ハイパボリック コサイン hyperbolic cosine	[H] [C] [S] X [↵] $\frac{e^{x}+e^{-x}}{2}$	
	$\tanh x$	HTN	ハイパボリック タンジェント hyperbolic tangent	[H] [T] [N] X [↵] $\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
逆双曲線関数	$\sinh^{-1}x$	AHS	アークハイパボリック サイン arc hyperbolic sine	[A] [H] [S] X [↵] $\log_e (x+\sqrt{x^2+1})$ x は任意の数	
	$\cosh^{-1}x$	AHC	アークハイパボリック コサイン arc hyperbolic cosine	[A] [H] [C] X [↵] $\pm\log_e (x+\sqrt{x^2-1})$ $x \geqq 1$	
	$\tanh^{-1}x$	AHT	アークハイパボリック タンジェント arc hyperbolic tangent	[A] [H] [T] X [↵] $\log_e\sqrt{\frac{1+x}{1-x}}$　$-1 < x < 1$	

（注）Xについては、60進数の度・分・秒の記号で指定することもできます。

(4)関数入力の方法 [4]

項目 通常の表現	本機の表現	読みかた	入力方法と備考
60進数(度､分､秒)→10進数(度)の変換	DEG	ディグリー to decimal degree	[→DEG] X [↵] またはX [→DEG] （DAns 機能）
● 度、分、秒の記号については、次の“(5) 60進数の計算について”を参照してください。			
10進数(度)→60進数(度､分､秒)の変換	DMS	ディ・エム・エス to degrees, minutes, second	[2nd F] [→DMS] X [↵] またはX [2nd F] [→DMS] (DAns 機能) ● 変換結果は、度、分、秒の記号で表示されます。
● 度、分、秒の記号については、次の“(5) 60進数の計算について”を参照してください。			
16進数→10進数の変換	&H	アンド・ヘキサデシマル & hexadecimal	[2nd F] [&] [H] h [↵] ただし、hは0～FFFFFFFFまでの16進数（A～Fはアルファベットキーで入力）
● 10進数→16進数の変換については、333ページのHEX$命令を参照してください。			
直交座標→極座標の変換 Z＝X＋Yi →Z＝(r，θ)	POL	ポーラ to polar coordinates (極)	● DEG､RAD､GRADのいずれかを指定 [2nd F] [→rθ] [(] X [,] Y [)] [↵] →rの値 [Z] [↵] →θの値 ここで、rの値を確認したいときは [Y] [↵] →rの値 θの値を確認したいときは [Z] [↵] →θの値
極座標→直交座標の変換 Z＝(r，θ) →Z＝X＋Yi	REC	レキュタンギュラー to rectangular coordinates (直角の、長方形)	● DEG､RAD､GRADのいずれかを指定 [2nd F] [→xy] [(] r [,] θ [)] [↵] →Xの値 [Z] [↵] →Yの値 ここで、Xの値を確認したいときは [Y] [↵] →Xの値 Yの値を確認したいときは [Z] [↵] →Yの値

（注）座標変換では、計算結果を変数YとZに入れますので、それまでY、Z（あるいはY＄、Z＄）に入っていた内容は消されます。

(5) 60進数（度、分、秒）計算について

60進数を使った度・分・秒計算ができます。

60進数の入力や表示には、数値の他に記号（°（度）、′（分）、″（秒））を使用します。

60進数（度、分、秒）→ 10進数（度）の変換（DEG）

度・分・秒の記号を使って変換するときは、VDEG命令（363ページ）を用います。

変換結果は数値で表示されます。

〈例〉25度45分18秒を10進数に変換

キー操作	表示部
[V] [D] [E] [G] [SHIFT] + ["] 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [SHIFT] + ["] [↵]	VDEG"25°45′18″"_ 25.755

10進数（度）→ 60進数（度、分、秒）の変換（DMS）

変換結果は度・分・秒の記号で表示されます。

〈例〉　123.6789度を60進数に変換

キー操作	表示部
123 [・] 6789 [2nd F] [→DMS]	123°40′44.04″（度 分 秒）
または [2nd F] [→DMS] 123 [・] 6789 [↵]	123°40′44.04″ 秒の端数（小数点以下）は、10進数になります。

（注）● 本機は、演算結果の値と進数を記憶していますので、ラストアンサー機能を用いたときは、演算結果の内容がそのまま呼び出されます。

上記の演算例のあとでは、([ANS] を押すと）１２３°４０′４４．０４″と表示されます。

● 演算結果が度・分・秒の記号で表示されたときは、[F↔E] の操作は無効です。

〈例〉

キー操作	表示部
25 [・] 4518 [→DEG]	25.755
[F↔E]（指数表示に変換）	2.5755E 01

● 度・分・秒の記号表示が対応できる範囲は $-10^4 < X < 10^4$ です。この範囲を超えたときは、"ERROR 33"と表示されます。

● その他の表示例

キー操作	表示部
0 [・] 001 [2nd F] [→DMS]	0°00′03.6″
1 [2nd F] [→DMS]	1°
0 [・] 1 [2nd F] [→DMS]	0°06′
9999 [・] 009 [2nd F] [→DMS]	9999°00′32.4″

度・分・秒の記号を使った演算例

① [2nd F] [→DMS] 1.23＋0.5 [↵] ⇒ 1.73

② [→DEG] 25.4518＋0.5 [↵] ⇒ 26.255

③ VDEG"25°45′18″"＋0.5 [↵] ⇒ 26.255

④　VDEG"２５°４５′１８″"＋
　[→DEG] ２５．４５１８＋
　[2nd F] [→DMS] １．２３[↵]　⇒　５２．７４
⑤　１°２０′＋２°２０′[↵]　⇒　３°４０′
⑥　３°３０′４５″＋６°４５′３６″[↵]　⇒　１０°１６′２１″
⑦　[→DEG] ２５．４５１８＋[2nd F] [→DMS] １．２３[↵]
　⇒　２６．９８５
⑧　３°４５′－１．６９[↵]　⇒　２．０６
⑨　３°３０′４５″×１０[↵]　⇒　３５°０７′３０″
⑩　１０÷３°３０′４５″[↵]　⇒　２．８４６９７５０８９

（注）従来機ＰＣ－Ｇ８５０Ｓでは、60進数に変換した値をそのまま演算に使用できません。
本機では、60進数に変換した値を演算に使用できますが、同じ演算でも演算結果が異なることがあります。

(6)取り扱う角度の単位の指定について

三角関数、逆三角関数および座標変換などの計算では、取り扱う角度の単位を正しく指定しておく必要があります。
マニュアル計算では、[2nd F] [DRG] の操作で指定できますので、度・ラディアン・グラードの指定をしたうえで計算を行ってください。なお、次の命令で指定することもできます。

角度単位	命令	表示シンボル	備考
度	ＤＥＧＲＥＥ[↵]	ＤＥＧ	直角を90で表す単位〔°〕
ラディアン	ＲＡＤＩＡＮ[↵]	ＲＡＤ	直角を $\frac{\pi}{2}$ で表す単位〔rad〕
グラード	ＧＲＡＤ[↵]	ＧＲＡＤ	直角を100で表す単位〔g〕

これは、ＢＡＳＩＣ言語でプログラムを組むときなどに使いますので、使いなれておいてください。

2．マニュアル計算における関数計算の操作方法と練習

1　2乗 [x^2]

例題を始める前に [CLS] を押して表示をクリアしてください。
数字が表示されていると、関数キーを押したとき、ダイレクトアンサー機能により計算を行ってしまいます。

	例題	キー操作	表示部
1	37^2	[x^2] 37 [↵] または 37 [x^2]	ＳＱＵ３７_ １３６９． １３６９．
2	$(87\times57)^2$	[x^2] [(] 87 [*] 57 [)] [↵] または 87 [*] 57 [↵] [x^2]	ＳＱＵ(８７＊５７)_ ２４５９１６８１． ２４５９１６８１．

	例題	キー操作	表示部
3	$\sqrt{5^2-4^2}$	[√] [(] [x^2] 5 [－] [x^2] 4 [)] [↵] または 5 [x^2] [－] [x^2] 4 [↵] [√]	ＳＱＲ(ＳＱＵ５－ＳＱＵ４)_ ３． ３．

練習問題　4　を行ってください。

2　平方根 [√]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	$\sqrt{3}$	[√] 3 [↵] または 3 [√]	ＳＱＲ３_ １．７３２０５０８０８ １．７３２０５０８０８
2	$\sqrt{25+86}\times\sqrt{37}$	[√] [(] 25 [＋] 86 [)] [*] [√] 37 [↵] または 25 [＋] 86 [↵] [√] [*] [√] 37 [↵]	ＳＱＲ(２５＋８６)＊ＳＱＲ３７_ ６４．０８５８７９８８ ６４．０８５８７９８６

（注）計算のしかたにより誤差が生じます。

誤差について

本機の内部では有効数字12桁で計算し、表示するときに11桁目を四捨五入して10桁で表示します。
計算を途中で区切ったときは、有効数字10桁にまるめた数で以後の計算を行います。
そのため、連続計算した場合と、計算の途中で区切った場合とでは誤差が生じます。

練習問題　5　を行ってください。

3 3乗 [2nd F] [x^3]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	3^3	[2nd F] [x^3] 3 [↵] または 3 [2nd F] [x^3]	CUB3_ 27. 27.
2	2^3+3^3	[2nd F] [x^3] 2 [+] [2nd F] [x^3] 3 [↵] または 2 [2nd F] [x^3] [+] [2nd F] [x^3] 3 [↵]	CUB2+CUB3_ 35. 35.

練習問題	6

を行ってください。

4 立方根 [2nd F] [$\sqrt[3]{\ }$]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	$\sqrt[3]{125}$	[2nd F] [$\sqrt[3]{\ }$] 125 [↵] または 125 [2nd F] [$\sqrt[3]{\ }$]	CUR125_ 5. 5.
2	$\sqrt[3]{(25+38)(96-57)}$	[2nd F] [$\sqrt[3]{\ }$] [(] [(] 25 [+] 38 [)] [*] [(] 96 [−] 57 [)] [)] [↵] または 25 [+] 38 [↵] [*] [(] 96 [−] 57 [)] [↵] [2nd F] [$\sqrt[3]{\ }$]	CUR((25+38)*(96−57))_ 13.49382434 13.49382434

練習問題	7

を行ってください。

5 逆数 [$1/x$]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	$\frac{1}{125}$	[$1/x$] 125 [↵] または 125 [$1/x$]	RCP125_ 0.008 0.008
2	$\frac{1}{\sqrt{185}}$	[$1/x$] [(] [$\sqrt{\ }$] 185 [)] [↵] または 185 [$\sqrt{\ }$] [$1/x$]	RCP(SQR185)_ 0.073521462 0.073521462
3	$\frac{1}{\frac{1}{6}+\frac{1}{7}}$	[$1/x$] [(] [$1/x$] 6 [+] [$1/x$] 7 [)] [↵] または 6 [$1/x$] [+] [$1/x$] 7 [↵] [$1/x$]	RCP(RCP6+RCP7)_ 3.230769231 3.23076923

練習問題	8

を行ってください。

6 べき乗 [y^x∧]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	8^5	8 [y^x∧] 5 [↵]	8∧5_ 32768.
2	$8^{5.37}$	8 [y^x∧] 5.37 [↵]	8∧5.37_ 70728.30171
3	5×7^4	5 [*] 7 [y^x∧] 4 [↵]	5*7∧4_ 12005.
4	$2.7^{3.4}+4.3^{2.5}$	2.7 [y^x∧] 3.4 [+] 4.3 [y^x∧] 2.5 [↵]	2.7∧3.4+4.3∧2.5_ 67.62611161
5	4^{-2}　$\left(\frac{1}{4^2}\right)$	4 [y^x∧] [−] 2 [↵]	4∧−2_ 0.0625

練習問題 9 10 11 を行ってください。

7 べき乗根 y^x

平方根√￣（$\frac{1}{2}$乗）、立方根∛￣（$\frac{1}{3}$乗）を一般化すると、$\sqrt[x]{Y} = Y^{\frac{1}{x}}$ の関係になります。つまりこれは、べき乗の変形と考えられます。

べき乗根の公式

m、n、pは正の整数、a＞0、b＞0とするとき、以下の式がなりたちます。

1. $\sqrt[n]{a} = a^{\frac{1}{n}}$
2. $(\sqrt[n]{a})^n = a$
3. $(\sqrt[n]{a})^m = \sqrt[n]{a^m} = a^{\frac{m}{n}}$
4. $\sqrt[np]{a^{mp}} = a^{\frac{mp}{np}} = a^{\frac{m}{n}}$
5. $\sqrt[n]{\sqrt[m]{a}} = (\sqrt[n]{a^{\frac{1}{m}}}) = a^{\frac{1}{mn}} = \sqrt[mn]{a}$
6. $\sqrt[n]{a} \cdot \sqrt[n]{b} = \sqrt[n]{a \cdot b}$
7. $\frac{\sqrt[n]{a}}{\sqrt[n]{b}} = \sqrt[n]{\frac{a}{b}}$

練習問題 12 を行ってください。

8 階乗 2nd F n!
順列 nPr　　組み合わせ 2nd F nCr

(1)階乗

nから１までの整数をかけ合わせて作った値をn!と表し、nの階乗と言います。

n!＝n×（n－1）×（n－2）×………×3×2×1

たとえば、

5!＝5×4×3×2×1＝120になります。

例題を始める前に CLS を押して表示をクリアしてください。

	例題	キー操作	表示部
1	5!	2nd F n! 5 ↵ または 5 2nd F n!	FACT5_ 120. 120.
2	8!×5!	2nd F n! 8 * 2nd F n! 5 ↵	FACT8＊FACT5_ 4838400.

(2)順列・組み合わせ

相異なるn個のものからr個を取って１組としたものを、n個のものからr個を取る組み合わせ（Combination）といい、その組み合わせの数を$_nC_r$で表します。

r個を取り出した上で、さらに取り出したr個のものを１列に順序づけて並べたものを、n個のものからr個を取る順列（Permutation）といい、その順列の数を$_nP_r$で表します。

順　列　$_nP_r = \frac{n!}{(n-r)!}$　（r≦n）

組み合わせ　$_nC_r = \frac{n!}{r!(n-r)!}$　ここで0!＝1と定義します。

	例題	キー操作	表示部
1	・1から10のボールから7つのボールを取り出して並べるしかたの数は $_{10}P_7 = \frac{10!}{(10-7)!}$	$_nP_r$ (10 , 7) ↵ または 2nd F n! 10 / 2nd F n! (10 − 7) ↵	NPR(10,7)_ 604800. 604800.
2	・10チームでリーグ戦を行うときの全試合数は $_{10}C_2 = \frac{10!}{2!(10-2)!}$	2nd F $_nC_r$ (10 , 2) ↵ または 2nd F n! 10 / (2nd F n! 2 * 2nd F n! (10 − 2)) ↵	NCR(10,2)_ 45. 45.

練習問題 13 14 を行ってください。

9 常用対数 log

a^m＝Mのとき、指数mはaを底とするMの対数といい、m＝$\log_a M$と表します。

このときMを対数mの真数といいます。

真数は常に正です。特に、底が10の対数を常用対数と呼び、底を略してlog Mと表します。

対数の基本的性質

1. $\log 1 = 0$
2. $\log 10 = 1$
3. $\log 100 = 2$
4. $\log(M \cdot N) = \log M + \log N$
　（対数で表すと、掛け算がたし算になります）
5. $\log \frac{M}{N} = \log M - \log N$
　（対数で表すと、わり算が引き算になります）
6. $\log M^P = P \cdot \log M$
7. $\log \sqrt[n]{M^m} = \log M^{\frac{m}{n}} = \frac{m}{n} \log M$
8. $\log_a b = \frac{\log b}{\log a}$
9. $\log_b a = \frac{1}{\log_a b}$　（底の変換公式）
10. $10^{\log M} = M$

例題を始める前に [CLS] を押して表示をクリアしてください。

	例　　題	キ　ー　操　作	表　示　部
1	$\log 1000$	[log] 1000 [↵] または 1000 [log]	LOG1000_ 3. 3.
2	$\log \sqrt{15}$	[log] [√] 15 [↵]	LOGSQR15_ 0.588045629
3	$\log \frac{9}{15}$ ※	[log] [(] 9 [/] 15 [)] [↵]	LOG(9／15)_ −0.221848749

※（　）を忘れると $\log 9 \div 15$ の計算になります。

練 習 問 題	15

を行ってください。

10 常用指数 [2nd F] [10^x]

10のべき乗を求めるときに用います。
これは $\log x$ の逆関数で、対数の真数を求めることになります。（58ページ逆三角関数の項参照）
例題を始める前に [CLS] を押して表示をクリアしてください。

	例　　題	キ　ー　操　作	表　示　部
1	10^3	[2nd F] [10^x] 3 [↵]	TEN3_ 1000.
2	$10^{7.4} + 10^{8.3}$	[2nd F] [10^x] 7.4 [+] [2nd F] [10^x] 8.3 [↵]	TEN7.4+TEN8.3_ 224645095.8
3	$\log N = 1.5$ ⬇ $N = 10^{1.5}$	[2nd F] [10^x] 1.5 [↵]	TEN1.5_ 31.6227766

練 習 問 題	16

を行ってください。

11 自然対数 [ln]

10を底とする対数が常用対数であるのに対し、

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots = 2.718281828459\cdots\cdots$$

を底とする対数を、自然対数またはネーピアの対数といいます。

自然対数と常用対数の関係

$$\log x = \frac{\ln x}{\ln 10} = 0.434294481 \times \ln x$$

例題を始める前に [CLS] を押して表示をクリアしてください。

	例　　題	キ　ー　操　作	表　示　部
1	$\ln 10$	[ln] 10 [↵] または 10 [ln]	LN10_ 2.302585093 2.302585093
2	$\ln 18 + \ln 15$	[ln] 18 [+] [ln] 15 [↵]	LN18+LN15_ 5.598421959
3	$\ln (18 \times 15)$	[ln] [(] 18 [*] 15 [)] [↵]	LN(18＊15)_ 5.598421959
4	$\ln 16 - \ln 9$	[ln] 16 [−] [ln] 9 [↵]	LN16−LN9_ 0.575364144
5	$\ln\left(\frac{16}{9}\right)$	[ln] [(] 16 [/] 9 [)] [↵]	LN(16／9)_ 0.575364144

12 自然指数 2nd F e^x

この関数は ln x の逆関数（真数）を求める関数です。（逆三角関数の項参照）
例題を始める前に CLS を押して表示をクリアしてください。

	例　題	キー操作	表示部
1	e^1	2nd F e^x 1 ↵ または 1 2nd F e^x	EXP1_ 2.718281828 2.718281828
2	$e^{2.845}$	2nd F e^x 2.845 ↵	EXP2.845_ 17.20155867
3	$e^{-6.41}\times e^{8.65}$	2nd F e^x (−) 6.41 (*) 2nd F e^x 8.65 ↵	EXP−6.41*EXP8.65_ 9.393331288

2nd F 10^x、2nd F e^x、2nd F Exp の違いについて

2nd F 10^x	常用指数	10のべき乗を求めます	数式例 ① $10^{1.5}$ ② $10^{7.4}$ ③ $10^{-1.3\times6.4}$
2nd F e^x	自然指数	ｅのべき乗を求めます	数式例 ① e^1 ② $e^{-2.5}$ ③ $e^3\times e^{8.5}$
2nd F Exp	指数指定	指数部を置数するときに使います 指数は±(0〜99)の整数です。 31ページ参照	置数例 ① 2.5×10^3 2.5 2nd F Exp 3 と置数 (2.5E 3 と表示されます) ② 2×10^{-3} 2 2nd F Exp − 3 と置数 (2 E − 3 と表示されます)

練習問題 17 を行ってください。

13 三角関数 sin cos tan

正弦（sin θ）、余弦（cos θ）、正接（tan θ）を求めます。
右の図のように、xy 平面上で、動径ＯＰが始線ＯＸ（x 軸の正方向）となす角を θ とし、動径上の点Ｐの座標を (x, y)、$\overline{OP}=\sqrt{x^2+y^2}=r$ とすると、次のような関係がなりたちます。

このように考えるとわかりやすい。

sin は $\frac{高さ}{斜辺}$ で求められますから
Ｓを筆記体で書くと $\mathcal{S}$ となりますので
斜辺/高さ で $\frac{y}{r}$ になるという意味です。
Cos の場合もＣを∠と考えて 斜辺/底辺 で $\frac{x}{r}$
Tan の場合もＴを⌋と考えて 高さ/底辺 で $\frac{y}{x}$

■角度の単位

半径に等しい長さの弧に対する中心角は、円の大きさに関係なく一定です。
この一定の角を単位１（ラディアン）として、角を測る方法が弧度法です。
弧度法では、単位の名称を略して、角の大きさを無名数で表すのが普通です。
弧度法に対して、度を単位として角を測る方法（１直角＝90°、1°＝60′、1′＝60″）を60分法といいます。
また、この90°を100^g（グラード）として測る方法もあります。

60分法		弧度法		グラード法
ディグリー		ラディアン		グラード
DEG		RAD		GRAD
90°	←→	π/2	←→	100^g
180°	←→	π	←→	200^g
360°	←→	2π	←→	400^g

弧AB＝半径ｒ

(2)三角関数の計算を行う前に

三角関数の計算を行うときは、角度単位の指定をしてください。

まず、与えられた計算式において

1．60分法で求めるのか
2．弧度法で求めるのか
3．グラードで求めるのか

これをまず確認してください。そして、

① 60分法（ディグリー）で求めるなら、表示部の右側に小さな英字でＤＥＧが出ていることを確認してください。もしＤＥＧが出ていないならＤＥＧという英字が出るまで [2nd F] [DRG] を繰り返してください。（ＤＥＧ表示は、電源ＯＮ時に必ず表示されます。）

② 弧度法（ラディアン）で求めるなら、表示部にＲＡＤが出るまで [2nd F] [DRG] を繰り返してください。

③ グラード法（グラード）で求めるなら、表示部にＧＲＡＤが出るまで [2nd F] [DRG] を繰り返してください。

ＧＲＡＤ（グラード）について

ＧＲＡＤはヨーロッパの測量関係など特殊な業務で使用されている角度です。
日本ではあまりなじみがありませんが、直角が100^gと考えやすい数値のため、計算が楽になります。

■主な角の三角関数

60分法	0°	30°	45°	60°	90°	120°	135°	150°	180°	270°	360°
弧度法 / 三角関数	0	$\frac{\pi}{6}$	$\frac{\pi}{4}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	$\frac{3\pi}{4}$	$\frac{5\pi}{6}$	π	$\frac{3\pi}{2}$	2π
$\sin\theta$	0	$\frac{1}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{\sqrt{3}}{2}$	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0	-1	0
$\cos\theta$	1	$\frac{\sqrt{3}}{2}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	0	$-\frac{1}{2}$	$-\frac{1}{\sqrt{2}}$	$-\frac{\sqrt{3}}{2}$	-1	0	1
$\tan\theta$	0	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	∞	$-\sqrt{3}$	-1	$-\frac{1}{\sqrt{3}}$	0	∞	0
図による角度と符号											

[1] ＳＩＮ関数

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	$\sin 60°$	表示部にＤＥＧを表示させます [sin] 60 [↵] または 60 [sin]	ＳＩＮ６０_ ０.８６６０２５４０３ ０.８６６０２５４０３
2	$\sin\frac{\pi}{3}$	表示部にＲＡＤを表示させます [sin] [(] [π] [／] 3 [)] [↵] または [π] [／] 3 [↵] [sin]	ＳＩＮ(ＰＩ／３)_ ０.８６６０２５４０３ ０.８６６０２５４０３
3	$\sin(0.5\pi+2.4)$	表示部にＲＡＤを表示させます [sin] [(] 0.5 [*] [π] [+] 2.4 [)] [↵]	ＳＩＮ(０.５＊ＰＩ＋２.４)_ －０.７３７３９３７１５
4	$\sin^2 30°$	表示部にＤＥＧを表示させます [sin] 30 [y^x∧] 2 [↵] または 30 [sin] [y^x∧] 2 [↵]	ＳＩＮ３０∧２_ ０.２５ ０.２５
5	$\sin 18° 30'$	表示部にＤＥＧを表示させます [sin] [→DEG] 18.30 [↵] または [→DEG] 18.30 [↵] [sin] または 18.30 [→DEG] [sin]	ＳＩＮＤＥＧ１８.３０_ ０.３１７３０４６５６ ０.３１７３０４６５６ ０.３１７３０４６５６
6	$\sin 250^g$	表示部にGRADを表示させます [sin] 250 [↵] または 250 [sin]	ＳＩＮ２５０_ －０.７０７１０６７８１ －０.７０７１０６７８１
7	$\sin 25° 45' 18''$	表示部にＤＥＧを表示させます [sin] 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [↵] または 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [sin]	ＳＩＮ２５°４５′１８″_ ０. ４３４５２３８５６ ０. ４３４５２３８５６

角度指定について

三角関数の計算を行う場合は、ＤＥＧ、ＲＡＤ、ＧＲＡＤの角度指定が必要です。角度指定をまちがうと正しい結果が得られません。角度指定は [2nd F] [DRG] を押して切り替えます。

[2] ＣＯＳ関数

例題を始める前に [CLS] を押して表示をクリアしてください。

	例　　題	キ　ー　操　作	表　示　部
1	$4 \times \cos 25°$	表示部にＤＥＧを表示させます 4 [*] [cos] 25 [↵]	４＊ＣＯＳ２５_ ３．６２５２３１１４８
2	$\cos \frac{\pi}{5}$	表示部にＲＡＤを表示させます [cos] [(] [π] [/] 5 [)] [↵]	ＣＯＳ(ＰＩ／５)_ ０．８０９０１６９９４
3	$\cos(473^g + 168^g)$	表示部にGRADを表示させます [cos] [(] 473 [+] 168 [)] [↵]	ＣＯＳ(４７３＋１６８)_ −０．７９９６８４６５８
4	$\cos 25° \ 45' \ 18''$	表示部にＤＥＧを表示させます [cos] 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [↵] または 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [cos]	ＣＯＳ２５° ４５′ １８″_ ０．９００６６０３２３ ０．９００６６０３２３

[3] ＴＡＮ関数

例題を始める前に [CLS] を押して表示をクリアしてください。

	例　　題	キ　ー　操　作	表　示　部
1	$\tan 19.7°$	表示部にＤＥＧを表示させます [tan] 19.7 [↵]	ＴＡＮ１９．７_ ０．３５８０５１８３７
2	$\tan(24° 8' 55'' + 37° 19' 23'')$	表示部にＤＥＧを表示させます [tan] [(] [→DEG] 24.0855 [+] [→DEG] 37.1923 [)] [↵]	ＴＡＮ(ＤＥＧ２４．０８５５＋ＤＥＧ ３７．１９２３)_ １．８３９６００９１５
3	$\tan(\pi + \frac{3}{8}\pi)$	表示部にＲＡＤを表示させます [tan] [(] [π] [+] 3 [/] 8 [*] [π] [)] [↵]	ＴＡＮ(ＰＩ＋３／８＊ＰＩ)_ ２．４１４２１３５６２
4	$\tan 25° \ 45' \ 18''$	表示部にＤＥＧを表示させます [tan] 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [↵] または 25 [2nd F] [°] 45 [2nd F] [′] 18 [2nd F] [″] [tan]	ＴＡＮ２５° ４５′ １８″_ ０．４８２４５０３１４ ０．４８２４５０３１４

練 習 問 題　18　を行ってください。

[14] 逆三角関数 [2nd F] [sin⁻¹]、[2nd F] [cos⁻¹]、[2nd F] [tan⁻¹]

逆三角関数とは、三角関数 sin、cos、tanのそれぞれの逆関数です。

逆関数とは、$y = x$ という一次関数に対して対称になる関数をいいます。

常用対数 ←（互いに逆関数の関係）→ 常用指数

自然対数 ←（互いに逆関数の関係）→ 自然指数

これと同じように、

三角関数 ←（互いに逆関数の関係）→ 逆三角関数

それぞれ $x = y$ に対して対称な関数となります

$y = \sin x$ ←→ $x = \sin^{-1} y$ （アークサインワイと読む）

$y = \cos x$ ←→ $x = \cos^{-1} y$ （アークコサインワイと読む）

$y = \tan x$ ←→ $x = \tan^{-1} y$ （アークタンジェントワイと読む）

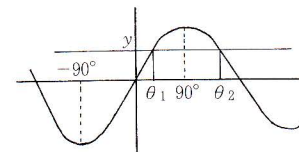
逆三角関数の計算は、三角関数と同様にＤＥＧ、ＲＡＤ、ＧＲＡＤの指定が必要です。

今、上の式の x を θ とおけば（$x = \theta$）

三 角 関 数	逆三角関数	逆三角関数の値の範囲		
		Ｄ　Ｅ　Ｇ	Ｒ　Ａ　Ｄ	Ｇ　Ｒ　Ａ　Ｄ
$y = \sin\theta$	$\theta = \sin^{-1} y$	$-90° \leqq \theta \leqq 90°$	$-\frac{\pi}{2} \leqq \theta \leqq \frac{\pi}{2}$	$-100^g \leqq \theta \leqq 100^g$
$y = \cos\theta$	$\theta = \cos^{-1} y$	$0° \leqq \theta \leqq 180°$	$0 \leqq \theta \leqq \pi$	$0^g \leqq \theta \leqq 200^g$
$y = \tan\theta$	$\theta = \tan^{-1} y$	$-90° \leqq \theta \leqq 90°$	$-\frac{\pi}{2} \leqq \theta \leqq \frac{\pi}{2}$	$-100^g \leqq \theta \leqq 100^g$

- 三角関数　$\sin\theta$、$\cos\theta$ の値は、すべて－１から＋１の値にありますから、それ以外の値で $\sin^{-1} y$、$\cos^{-1} y$ の計算はできません。

メモ　逆三角関数の値の範囲について

$\sin\theta$ の関数は右図のようになります。
$\sin^{-1} y$ は y のときの θ を求めますが、右図の y の場合、計算機は θ_1 か θ_2 かの判断ができません。そのため、計算機では求められる θ が１つしか存在しない－90°～90° の範囲に限定しています。θ_2 を知りたいときは、$180° - \theta_1$ を計算してください。

[1] sin⁻¹、cos⁻¹、tan⁻¹

例題を始める前に [CLS] を押して表示をクリアしてください。

	例　　題	キ　ー　操　作	表　示　部
1	$\sin^{-1}0.5$ 角度単位（ＤＥＧ）	表示部にＤＥＧを表示させます [2nd F] [sin⁻¹] 0.5 [↵] または 0.5 [2nd F] [sin⁻¹]	 ＡＳＮ０．５_ ３０． ３０．
2	$\cos^{-1}0.628$ （ＲＡＤ）	表示部にＲＡＤを表示させます [2nd F] [cos⁻¹] 0.628 [↵] または 0.628 [2nd F] [cos⁻¹]	 ＡＣＳ０．６２８_ ０．８９１８１５７７７ ０．８９１８１５７７７
3	$\tan^{-1}1$ （ＧＲＡＤ）	表示部にGRADを表示させます [2nd F] [tan⁻¹] 1 [↵] または 1 [2nd F] [tan⁻¹]	 ＡＴＮ１_ ５０． ５０．
4	$2\sin^{-1}0.785$ （ＤＥＧ）	表示部にＤＥＧを表示させます 2 [*] [2nd F] [sin⁻¹] 0.785 [↵] 度分秒で求める場合は、ここで [2nd F] [→DMS]	 ２＊ＡＳＮ０．７８５_ １０３．４４１３５６５ (103.4413565度) １０３°２６′２８．８８″ (103度26分28.88秒)
5	$\cos^{-1}0.43+$ $\cos^{-1}0.66$ （ＲＡＤ）	表示部にＲＡＤを表示させます [2nd F] [cos⁻¹] .43 [+] [2nd F] [cos⁻¹] .66 [↵]	 ＡＣＳ．４３＋ＡＣＳ．６６_ １．９７６２８１１１６
6	$\tan^{-1}\sqrt{\dfrac{1-0.6^2}{0.6}}$ （ＤＥＧ）	表示部にＤＥＧを表示させます [2nd F] [tan⁻¹] [√] [(] [(] 1 [−] .6 [yˣ] 2 [)] [/] .6 [)] [↵]	 ＡＴＮＳＱＲ((１−．６∧２) ／．６)_ ４５．９２４２８５８２

練 習 問 題	19	を行ってください。

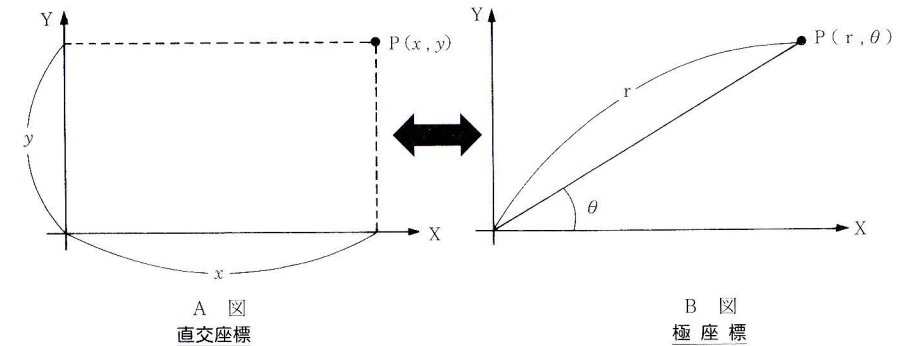
15 座標変換

座標変換とは、ベクトルや複素数において、x成分・y成分で表される式を大きさｒ・偏角θに変換することや、その逆に、大きさｒ・偏角θからx成分・y成分に変換することをいいます。

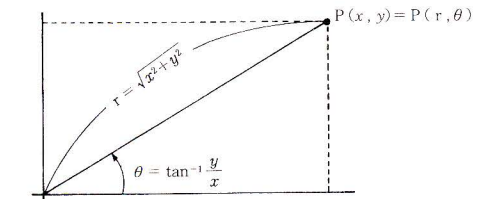
x成分、y成分として表されるものを、　**直交座標**　$Z=x+yi$
大きさｒ，偏角θとして表されるものを　**極座標**　$Z=(r,\theta)$
と、いいます。

この座標変換の場合も、三角関数同様、角度を扱いますので、角度指定ＤＥＧ、ＲＡＤ、ＧＲＡＤのいずれかを指定する必要があります。

Ａ　図
直交座標

Ｂ　図
極座標

Ａ図も、Ｂ図もＸＹ平面上の点Ｐを表現する方法です。表現方法は違いますが、点Ｐの位置を示していることにおいては同じ結果となります。

直交座標を極座標に変換するには

- 極座標のｒを求める　⟶　$r=\sqrt{x^2+y^2}$
- 偏角θを求める　⟶　$\theta=\tan^{-1}\dfrac{y}{x}$

極座標を直交座標に変換するには

- 直交座標のxを求める　⟶　$x=r\cos\theta$
- 直交座標のyを求める　⟶　$y=r\sin\theta$

[1] 直交座標 → 極座標変換 [2nd F] [→rθ]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	(3,8)を極座標に変換 (P＝3＋8i)	角度単位を“度”に指定します 表示部にＤＥＧを表示させます [2nd F] [→rθ] [(] 3 [,] 8 [)] [↵] [Z] [↵]	 POL(3, 8)_ 8.544003745 （ｒの値） Z_ 69.44395478 （θの値）
		〔解説〕 直交座標（３，８）は極座標に変換すると約（8.54，69.4°）です	
		ここでｒの値を確認したいときは [Y] [↵] これはｒの値が変数Ｙに収められていることを意味します 次にθの値を確認したいときは [Z] [↵] θの値は変数Ｚに収められていることを意味します さらにθの値は度で表されているので度・分・秒に変換したいときは [2nd F] [→DMS]	 Y_ 8.544003745 Z_ 69.44395478 69° 26′ 38. 24″ (69度26分38.24秒)
2	(３，－８) を極座標に変換 (P＝３－８i)	表示部にＤＥＧを表示させます [2nd F] [→rθ] [(] 3 [,] [−] 8 [)] [↵] [Z] [↵]	 POL(3, −8)_ 8.544003745 （ｒの値） Z_ −69.44395478 （θの値）
		〔解説〕 直交座標（３，－８）は極座標に変換すると約（8.54，－69.4°）です	
3	(－３，－８) を極座標に変換 (P＝－３－８i)	表示部にＤＥＧを表示させます [2nd F] [→rθ] [(] [−] 3 [,] [−] 8 [)] [↵] [Z] [↵]	 POL(−3, −8)_ 8.544003745 （ｒの値） Z_ −110.5560452 （θの値）

[2] 極座標→直交座標変換 [2nd F] [→xy]

例題を始める前に [CLS] を押して表示をクリアしてください。

	例題	キー操作	表示部
1	(2，30°)を直交座標に変換 P＝(2，30°)	表示部にＤＥＧを表示させます [2nd F] [→xy] [(] 2 [,] 30 [)] [↵] [Z] [↵]	 REC(2, 30)_ 1.732050808 (xの値) Z_ 1. (yの値)
		〔解説〕 極座標（2, 30°）は直交座標に変換すると約（1.73, 1）です ● xの値を再確認するときは [Y] [↵] 1.732050808 xの値は変数Ｙに収められていることを意味します ● yの値を再確認するときは [Z] [↵] 1. yの値は変数Ｚに収められていることを意味します	
2	(2,－30°)を直交座標に変換 P＝(2,－30°)	表示部にＤＥＧを表示させます [2nd F] [→xy] [(] 2 [,] [−] 30 [)] [↵] [Z] [↵]	 REC(2, −30)_ 1.732050808 Z_ −1.
3	(2, $\frac{\pi}{3}$) を直交座標に変換 P＝(2, $\frac{\pi}{3}$)	表示部にＲＡＤを表示させます [2nd F] [→xy] [(] 2 [,] [π] [/] 3 [)] [↵] [Z] [↵]	 REC(2, PI／3)_ 1. Z_ 1.732050808

練習問題 20 を行ってください。

[16] 統計計算

たくさんのデータをひとつのまとまった資料にするためには、統計的な処理が必要です。

本機は統計モードで、一変数統計計算、二変数統計計算ができます。

一変数統計計算は、たとえばテストの点数のような１種類のデータを用いて、その平均値、標準偏差などの統計量を求めることができます。

二変数統計計算は、たとえば身長と体重のように関連がある（と予想される）２種類のデータを用いて、それぞれのデータの平均や標準偏差を求めたり、２種類のデータの相関関係を調べたり、また一次回帰線による推定を行うことができます。

(1)統計モードの設定と解除の方法

1．統計モードにするときは

ＲＵＮ（またはＰＲＯ）モードで

[2nd F] [STAT]

と押します。統計モードになり、右の一変数／二変数選択画面が表示されます。

```
***** トウケイ ブンセキ *****

  1：1ヘンスウ トウケイ  (x)          STAT
  2：2ヘンスウ トウケイ  (x, y)        CAPS
                                      DEG
バンゴウ ヲ エランデ クダサイ.
```

なお、統計モードでは、画面右側に“ＳＴＡＴ”が点灯します。

この選択画面のとき、[1] で一変数統計計算、[2] で二変数統計計算が選べます。

2．統計モードを解除するときは

[2nd F] [STAT]

と押します。ＲＵＮモードになります。

(2)一変数統計計算

1．一変数統計計算で求める統計量

一変数統計計算では、次の統計量を求めることができます。

・サンプル数　n

・サンプルの総和　Σx

・サンプルの２乗の和　Σx^2

・平均　$\bar{x}=\dfrac{\Sigma x}{n}$

・サンプルの標準偏差　$s=\sqrt{\dfrac{\Sigma x^2-n\bar{x}^2}{n-1}}$

母集団より抽出されたサンプルデータから、母集団の標準偏差を推定する場合に使用します。

・母標準偏差　$\sigma=\sqrt{\dfrac{\Sigma x^2-n\bar{x}^2}{n}}$

母集団のすべてをサンプルデータとして、その標準偏差を求める場合、またはサンプルを母集団とみなして、その標準偏差を求める場合に使用します。

2．一変数統計計算の選択

[2nd F] [STAT] と押して統計モードにした後、画面に従って [1] を押せば一変数統計計算が選択され、処理選択画面になります。

[2nd F] [STAT]

```
***** トウケイ ブンセキ *****

  1：1ヘンスウ トウケイ  (x)
  2：2ヘンスウ トウケイ  (x, y)

バンゴウ ヲ エランデ クダサイ.
```

[1]

```
*** ショリ ***        (x)

1：ニュウリョク      2：サクジョ/クリア
3：ブンセキ          4：プリンタ

バンゴウ ヲ エランデ クダサイ.
```

処理選択画面では、それぞれ次の処理を選ぶことができます。

[1] ……入力　　データの入力を行うときに選びます。

[2] ……削除／クリア　　まちがったデータを入力した場合などに、そのデータを削除するときや、新たに統計計算を始める場合に、前に計算した統計量をすべて消去するときに選びます。

[3] ……分析　　統計量を求めるときに選びます。

[4] ……プリンタ　　統計量を印字するときに選びます。ただし、プリンタが接続されているときにのみ選ぶことができます。プリンタが接続されていないときは [4] を押しても何も変わりません。

なお、一つ前の選択画面に戻すときは BREAK [ON] を押します。

3．データの入力方法

処理選択画面で [1] を押せばデータ入力画面になり、データが入力できます。

[1]

①データを１つずつ入力する場合は

データ [↵]

と押します。

②負のデータを入力する場合は

[(−)] データ [↵]

と押します。

③同じデータが複数個ある場合は

データ [,] 個数 [↵]

と押します。

データをすべて入力したら BREAK [ON] を押します。処理選択画面に戻ります。

［補足］

統計では、同じデータが何個かある場合は“度数”という言葉を使います。

たとえば、同じデータが３個ある場合は“度数３”のように表します。

4．統計量を求める方法

処理選択画面で [3] を押せば、次の分析画面になります。

[3]

```
** ブンセキ **          (x)

1：n     2：Σx     3：Σx²    4：x̄
5：s     6：σ

バンゴウ ヲ エランデ クダサイ.
```

[1] ～ [6] で次の統計量を求めることができます。

[1] ……n　　サンプル数

[2] ……Σx　　サンプルの総和

[3] ……Σx^2　　サンプルの２乗の和

[4] ……$\bar{x}$　　平均値

[5] ……s　　サンプルの標準偏差

[6] ……σ　　母標準偏差

処理選択画面に戻すときは、BREAK [ON] を押します。

5．新たに統計計算を行うときは（前の計算内容を消すときは）

次の２つの方法があります。

①いったん統計モードを解除し、改めて統計モードにすれば消去されます。（統計モードを解除したとき、統計量の一部は変数に保存されています。くわしくは69ページを参照してください。）

②削除／クリア機能で消去します。まず、処理選択画面にして次のようにキーを押します。

[2]
削除／クリア選択画面になります。

```
** サクシ゛ョ／クリア **

 1：テ゛ータ サクシ゛ョ
 2：オール クリア

ハ゛ンコ゛ウ ヲ エランテ゛ クタ゛サイ.
```

[2]
消去確認画面になります。

```
* オール クリア *

1：YES
2：NO

ハ゛ンコ゛ウ ヲ エランテ゛ クタ゛サイ.
```

[1]　前の計算内容が消去され、処理選択画面に戻ります。
[2]を押した場合は、計算内容を消去せずに処理選択画面に戻ります。

■**例題**

ある試験の点数を、ランダムに選び出した35人について見た場合、右のようになりました。

これより平均値、標準偏差を求めなさい。

No.	点数	人数	No.	点数	人数
1	30	1	5	70	8
2	40	1	6	80	9
3	50	4	7	90	5
4	60	5	8	100	2

キー操作	表示部
・統計モードにします。　[2nd F] [STAT]	
・一変数統計計算を選びます。　[1]	
・“入力”を選びます。　[1]	
・データを入力します。	
30 [↵] 40 [↵]　50 [,] 4 [↵]	
60 [,] 5 [↵]　70 [,] 8 [↵]	
80 [,] 9 [↵]　90 [,] 5 [↵]	
100 [,] 2 [↵]	34：x＝100.,2.
これでデータの入力は終わりです。	36：x＝_
・処理選択画面に戻します。　[BREAK ON]	
・“分析”を選びます。　[3]	
・平均値を求めます。　[4]	$\bar{x}$＝　71.42857143
・サンプルの標準偏差を求めます。　[5]	s＝　16.47508942
・母標準偏差を求めます。　[6]	σ＝　16.23802542
・処理選択画面に戻します。　[BREAK ON]	

［補足］

サンプル数、総和、２乗の和を求めるときは、分析画面でそれぞれ[1]、[2]、[3]を押します。

途中結果として平均値や標準偏差などの統計量を求めた後、もう一度、処理選択画面で“入力”を選んでデータを入力すれば、続きのデータとして入力できます。

6．データの削除

データ入力でまちがったデータを入れた場合などに使用する機能です。

処理選択画面にして次のようにキーを押すとデータ削除画面になります。

[2]

```
** サクシ゛ョ／クリア **

 1：テ゛ータ サクシ゛ョ
 2：オール クリア

ハ゛ンコ゛ウ ヲ エランテ゛ クタ゛サイ.
```

[1]

```
* テ゛ータ サクシ゛ョ *
x＝_
```

この画面で、データ入力のときと同様の操作で、まちがったデータや削除したいデータを入力すれば削除されます。データ入力のときと同様に[,]を使って複数個のデータを削除することもできます。削除した後、[BREAK ON] [1]と押してデータ入力画面にしてから正しいデータを入力すれば、続きのデータとして入力できます。

［補足］

入力していない数値を削除したいデータとして入力することもできます。つまり、30と40の入力に対し、削除したいデータとして35を入力することも可能ですが、正しい結果は得られません。データを削除するときはご注意ください。

7．統計量の印字

CE-126P（プリンタ）をお持ちの場合は、データを入力した後、計算された統計量を印字できます。プリンタを本機に接続して電源を入れた後、データを入力します。次に処理選択画面で、[4]を押して“プリンタ”を選べば印字が開始されます。

```
*** ショリ ***      (x)

1：ニュウリョク    2：サクシ゛ョ／クリア
3：フ゛ンセキ      4：フ゜リンタ

ハ゛ンコ゛ウ ヲ エランテ゛ クタ゛サイ.
```

[4]

```
** インシ゛チュウ **
```

印字が終われば処理選択画面に戻ります。

〈印字例〉65ページの例題のデータが入っている場合

```
n=                  35.
Σx=               2500.
Σx²=            187800.
MEAN(x)=    71.42857143
s=          16.47508942
σ=          16.23802542
```

(3)二変数統計計算

二変数統計計算の基本的な操作方法は、一変数統計計算と同じです。先に一変数統計計算の説明をお読みください。

1．二変数統計計算で求める統計量

二変数統計計算では、次の統計量を求めることができます。

- n、Σx、Σx^2、$\bar{x}$ は一変数統計計算と同じ。sx、σx は一変数統計計算の s 、σ と同じ。
- Σy　サンプル（y）の総和
- Σy^2　サンプル（y）の2乗の和
- Σxy　サンプル（x，y）の積の和
- $\bar{y}$　$\bar{y}=\dfrac{\Sigma y}{n}$　サンプル（y）の平均値
- sy　$sy=\sqrt{\dfrac{\Sigma y^2-n\bar{y}^2}{n-1}}$　サンプル（y）から求める、母数を（n－1）としたときの標準偏差
- σy　$\sigma y=\sqrt{\dfrac{\Sigma y^2-n\bar{y}^2}{n}}$　サンプル（y）から求める、母数を（n）としたときの標準偏差
- a　$a=\bar{y}-b\bar{x}$　一次回帰線 y＝a＋b x の係数
- b　$b=\dfrac{Sxy}{Sxx}$　一次回帰線 y＝a＋b x の係数
- r　$r=\dfrac{Sxy}{\sqrt{Sxx\cdot Syy}}$　相関係数
- x′　$x'=\dfrac{y-a}{b}$　推定値（y の値から x の値を推定する）
- y′　$y'=a+bx$　推定値（x の値から y の値を推定する）

［補足］

$$Sxx=\Sigma x^2-\frac{(\Sigma x)^2}{n}\qquad Syy=\Sigma y^2-\frac{(\Sigma y)^2}{n}\qquad Sxy=\Sigma xy-\frac{\Sigma x\cdot\Sigma y}{n}$$

2．二変数統計計算の選択

[2nd F][STAT] と押して統計モードにした後、画面に従って [2] を押せば二変数統計計算が選択されて、処理選択画面になります。

3．データの入力方法

処理選択画面で [1] を押せばデータ入力画面になります。画面に従って、x，y のデータを入力してください。

①データが1個（1組）の場合

　データ x [↵] データ y [↵]

と押します。

②同じデータが複数個ある場合

　データ x [↵] データ y [,] 個数 [↵]

と押します。

③負数のデータは、それぞれのデータの前に [(−)] を押します。

データをすべて入力したら [ON]（BREAK）を押します。処理選択画面に戻ります。

4．統計量を求める方法

処理選択画面で [3] を押せば分析画面になります。二変数統計計算では分析画面が2画面あります。画面の切り替えは [▼] [▲] で行います。

処理選択画面で [3] を押した場合（“分析”の第1画面です）

```
**  ブンセキ  **          (x, y)    ↓

1：n      2：Σx     3：Σx²    4：x̄
5：sx     6：σx     7：Σy     8：Σy²

バンゴウ ヲ エランデ クダサイ.
```

[▼]（“分析”の第2画面です）

```
**  ブンセキ  **          (x, y)    ↑

1：Σxy  2：ȳ        3：sy     4：σy
5：a  6：b  7：r  8：x′       9：y′

バンゴウ ヲ エランデ クダサイ.
```

[▲] で前の（第1）画面に戻ります。

■例題

次の表は、ある地方の山桜の開花日（4月）と同地3月の平均気温の表です。

これより、一次回帰線 y＝a＋b x の係数 a 、b と相関係数 r を求め、3月の平均気温が9.1度の場合の開花日および4月10日に開花した年の3月の平均気温を推定します。

年	1	2	3	4	5	6	7	8
平均気温（x度）	6.2	7.0	6.8	8.7	7.9	6.5	6.1	8.2
開 花 日（y日）	13	9	11	5	7	12	15	7

キ　ー　操　作		表　示　部
・統計モードにします。	[2nd F] [STAT]	
・二変数統計計算を選びます。	[2]	
・“入力”を選びます。	[1]	
・データを入力します。		
6.2 [↵] 13 [↵] 7.0 [↵] 9 [↵]		
6.8 [↵] 11 [↵] 8.7 [↵] 5 [↵]		
7.9 [↵] 7 [↵] 6.5 [↵] 12 [↵]		
6.1 [↵] 15 [↵] 8.2 [↵] 7 [↵]		8：x＝8.2 y＝7.
これでデータの入力は終わりです。		9：x＝_
・処理選択画面に戻します。	BREAK [ON]	
・“分析”を選びます。	[3]	
・第２画面を呼び出します。	[▼]	
・係数ａを求めます。	[5]	a＝　34.4495101７
・係数ｂを求めます。	[6]	b＝　－3.425018839
・相関係数ｒを求めます。	[7]	r＝　－9.691068372E－01 (－0.9691068372)
・開花日を推定します。	[9]	x＝_
平均気温を入力	9.1 [↵]	y＝　3.281838734 （推定：４月３日ごろ開花）
・“分析”の第２画面に戻します。	BREAK [ON]	
・平均気温を推定します。	[8]	y＝_
開花日を入力	10 [↵]	x＝　7.13850385 （推定：３月の平均気温は約7.1℃）
統計モードを解除	[2nd F] [STAT]	

［補足］

統計計算の統計量のうち、次のものは変数Ｕ～Ｚに入れられ、統計モードを解除しても保持されています。したがって、ＲＵＮモードでも、この統計量を使って計算ができます。

	変　数	U	V	W	X	Y	Z
統計量	一変数統計	――	――	――	Σx^2	Σx	n
	二変数統計	Σy^2	Σy	Σxy	Σx^2	Σx	n

なお、この内容は統計モードになったときに消去されます。

練習問題 21 を行ってください。

第３章
算術代入計算

1．算術代入計算

今までは、数値と数値の計算のやりかたなどを練習してきましたが、複雑な数式や数値のたくさんあるような式を計算処理していくときに、どうしても数式（数値を代入するので）が長くなってしまいます。こんなとき、あらかじめ数値を変数キー（アルファベットキー）に記憶させておき、変数と変数の計算によって答えを求めることができます。

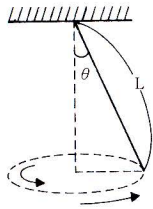
変数として使える文字

変数として使える文字は、[英文字]、[英文字] [英文字] の組み合わせ、[英文字] [数字] の組み合わせの３種類で、最高２文字分の長さまで使えます。

種　　類	例	
1. [英文字] （アルファベット）	[A]～[Z]	A＝1 B＝100 Z＝2.5
2. [英文字] [英文字]	[A]～[Z]	AA＝1.5 MM＝30.5 LX＝0.003
3. [英文字] [数　字]	[A]～[Z] [0]～[9]	A1＝6.28 G1＝9.8 T2＝358.2 X1＝3 Y2＝6

（注）● アルファベットの小文字は大文字と同じ扱いになります。変数としてアルファベットの小文字を使用しても大文字に変換されます。
● 変数Ｕ、Ｖ、Ｗ、Ｘ、Ｙ、Ｚは座標変換や統計計算で使用するため、変数として使うときには注意が必要です。

２．例題と解説

■例　題①

左図のような円すい振り子の周期Ｔは、糸の長さをＬ、糸の鉛直となす角をθとすると、

$$T = 2\pi\sqrt{\frac{L\cos\theta}{g}} \qquad g = 9.8〔m／sec^2〕$$

で求めることができます。Ｌ＝50cm　θ＝25°の場合の周期Ｔを求めなさい。

このような例題をプログラム化しますと、141ページの例題㉘のようなものが考えられます。

■解　説①

計算の内容	キ　ー　操　作	表　示　部
	表示部にＤＥＧを表示させます	
$g=9.8$ Ｇに9.8を代入します	[G][=]9.8　[↵]	9.8
$L=0.5$ Ｌに0.5を代入します	[L][=]0.5　[↵]	0.5
$\theta=25°$ θがありませんのでＳに25を代入します	[S][=]25　[↵]	25.
$T=2\pi\sqrt{\frac{L\cos\theta}{g}}$ 文字式として入力します	[T][=]2[*][π][*][√][(][L][*][cos][S][/][G][)]　[↵]	T=2*PI*SQR(L*COS S／G)_ 1.351106827 答　1.35秒
	計算の結果はＴに入っています [T]　[↵] と押せば呼び出されます	1.351106827

メモ　プレイバックについて

[↵]を押して計算を行った後に[▶]または[◀]を押すと、計算した式が呼び戻され、再度計算ができます。エラーになったときは、エラーになった位置にカーソルが表示されます。訂正して再び[↵]を押すと答が得られます。なお、カーソルは[▶]を押したときは式の先頭に、[◀]を押したときは式の最後に表示されます。

■例　題②

月の軌道を地球を中心とする円とみなし、地球の半径ｒ、月の周期をＴとすると、月の軌道半径は

$$R = \sqrt[3]{\frac{g\cdot r^2\cdot T^2}{4\pi^2}}$$

で与えられます。　地球の半径 $r=6.4\times10^6$〔m〕、月の周期Ｔ＝27日８時間としたときの月の軌道半径を求めなさい。

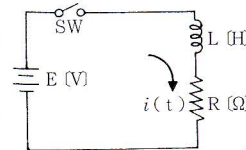
27日８時間は（27×24＋８）×60×60で秒に換算され、$T=2.3616\times10^6$〔sec〕になります。

また、ｇは例題①のようにｇ＝9.8〔m／sec^2〕とします。

■解　説②

計算の内容	キ　ー　操　作	表　示　部
$R=6.4\times10^6$	[R][=]6.4[2ndF][Exp]6　[↵]	R=6.4E6_ 6400000.
$T=2.3616\times10^6$	[T][=]2.3616[2ndF][Exp]6　[↵]	T=2.3616E6_ 2361600.
$G=9.8$	[G][=]9.8　[↵]	9.8
$RR=\sqrt[3]{\frac{G\cdot R^2\cdot T^2}{4\pi^2}}$ ここでＲは地球の半径で使っているので月の軌道半径の変数をＲＲとします	[R][R][=][2ndF][∛][(][G][*][(][R][yˣ]2[)][*][(][T][yˣ]2[)][/][(]4[*][(][π][yˣ]2[)][)][)]　[↵]	RR=CUR(G*(R∧2)*(T∧2)／(4*(PI∧2)))_ 384190235.2 〔m〕 答　384190km

このように、原式の文字に対する数値を記憶させてから文字式の演算をすると、後で式の確認も容易です。とても便利な使いかたのひとつです。

■例　題③

左図のようなＲＬ直列回路に流れる電流は、スイッチＳＷを閉じてからｔ秒後に

$$i(t) = \frac{E}{R}\left(1-e^{-\frac{R}{L}t}\right)$$

となります。

今、Ｅ＝4.5〔V〕Ｌ＝160〔mH〕Ｒ＝55〔Ω〕として

ｔ＝0.1〔ms〕における電流の値を求めなさい。

このような例題をプログラム化しますと、142ページの例題㉙のようなものが考えられます。

■解　説③

計算の内容	キ　ー　操　作	表　示　部
E＝4.5	[E] [=] 4.5 [↵]	4.5
L＝160〔mH〕＝160×10^{-3}	[L] [=] 160 [2nd F] [Exp] [−] 3 [↵]	0.16
R＝55	[R] [=] 55 [↵]	55.
T＝0.1〔ms〕＝0.1×10^{-3}	[T] [=] 0.1 [2nd F] [Exp] [−] 3 [↵]	0.0001
$I=\frac{E}{R}(1-e^{-\frac{R}{L}t})$	[I] [=] [E] [/] [R] [*] [(] 1 [−] [2nd F] [e^x] [(] [−] [R] [*] [T] [/] [L] [)] [)] [↵]	I＝E／R＊(1−EXP(−R＊T／L))_ 0.002764709 答　2.7647〔mA〕

■例　題④

物質の崩壊時間は　$t=\frac{1}{\lambda}\ln(1+\frac{Dt}{Pt})$ で求められます。

Dt＝ｔ時間後の同位原子数　Dt＝5×10^4個
Pt＝ｔ時間後の安定原子数　Pt＝3×10^6個
λ＝崩壊定数　λ＝20.5

このとき、この物質は何時間経過しているでしょうか。

■解　説④

計算の内容	キ　ー　操　作	表　示　部
Dt＝5×10^4	[D] [T] [=] 5 [2nd F] [Exp] 4 [↵]	50000.
Pt＝3×10^6	[P] [T] [=] 3 [2nd F] [Exp] 6 [↵]	3000000.
λ＝20.5	[V] [=] 20.5 [↵]	20.5
$t=\frac{1}{\lambda}\ln(1+\frac{Dt}{Pt})$	[T] [=] [1/x] [V] [*] [ln] [(] 1 [+] [D] [T] [/] [P] [T] [)] [↵]	T＝RCPV＊LN(1＋DT／PT)_ 0.000806307 答　0.000806307時間

練 習 問 題　22　を行ってください。

マニュアル計算の練習問題

● 解答は先生にお聞きください。（指導マニュアルに解答例を記載しています。）

練 習 問 題　1

下の例にならってキー操作の練習を行い、空欄に答えなさい。

	計　算　式	キ　ー　操　作	答
例	2＋3×4	2 [+] 3 [*] 4 [↵]	14
1	4.36−(5.25−4.83)		
2	$\frac{1.71\times2.43}{3.25\times1.35}$		
3	$6.48-\frac{6.87}{5.13+6.03-3.41}$		
4	$\frac{356}{243+638-475}$		
5	$\frac{4.37}{62.8+11.5\times37.8}$		
6	$\frac{28.4\times(0.75-2.49)+6.3}{2.52}$		
7	$\frac{-15\times3.54}{14.4}+\frac{18\times1.6}{3.74}$		
8	$\frac{-15\times0.73}{25.9}-\frac{38\times1.94}{3.74}$		

メモ　答えが正しくない？

カッコの指定を忘れていませんか？　2、3、4、5の問題ではカッコの指定が必要です。

練習問題 2

次の計算式を計算しなさい。ただし、式の中の↓のついているところで、区切って連続計算をする練習をしなさい。

	計算式	キー操作	答
1	$\frac{1.71\times2.43}{3.25\times1.35}$ ↙		
2	$\frac{6.48\times6.87}{5.13+6.03-3.41}$ ↙		
3	$\frac{15\times3.45}{14.4}+\frac{189\times0.58}{8.6}$ ↙		

練習問題 3

次の問題を小数点以下2桁で答えを求めなさい。

	計算式	キー操作	答
	まず、DIGIT（デジット）指定を行ってください。		
1	$4\times(-20)+5$		
2	$0.95+\frac{0.79}{3.6}$		
3	$\frac{8.5\times10^{20}}{6.24\times10^{18}}$		
4	$\frac{5\times1.8}{87.93+24.15}$		
5	$\frac{1}{1\times2}+\frac{2}{2\times3}+\frac{3}{3\times4}$		

USING（ユージング）指定でも計算してみましょう。

練習問題 4

	計算式	キー操作	答
1	$(263+185)^2$		
2	$15^2+38^2+73^2-56^2-24^2$		
3	$13^2\times28^2\div46^2\times89^2\div9^2$		
4	$(88^2+73^2)^2$		
5	$(2.85\times10^3)^2+(62.98\times10^2)^2$		

練習問題 5

	計算式	キー操作	答
1	$\sqrt{34}\times\sqrt{86}$		
2	$\sqrt{53+95}+\sqrt{0.84}$		
3	$18\times\pi\times\sqrt{\frac{843}{257}}$		
4	$\sqrt{12(12-5)(12-4)(12-3)}$		

練習問題 6

	計算式	キー操作	答
1	$(18+7)^3\times 0.5$		
2	$\left(\dfrac{5.8\times 10^2+3.8}{7.2\times\sqrt{105}}\right)^3$		
3	$(3.2+\sqrt{26.3\times 8.1})^3$		

練習問題 7

	計算式	キー操作	答
1	$\sqrt[3]{53\times 0.25+72\times 1.92}$		
2	$\sqrt[3]{3.2+\sqrt{26.3\times 8.1}}$		
3	$\sqrt[3]{\dfrac{980}{4}\left(\dfrac{6.4\times 10^8\times 2.3\times 10^6}{\pi}\right)^2}$		

練習問題 8

	計算式	キー操作	答
1	$\dfrac{1}{\sqrt{21\times 33}}$		
2	$\dfrac{1}{(87.93+24.15)\times(13.84-27.65)}$		
3	$\dfrac{1}{6.24\times 10^{-10}+3.85\times 10^{-10}+\sqrt[3]{1.3\times 10^{-30}}}$		
4	$\dfrac{1}{1\times 2}+\dfrac{1}{2\times 3}+\cdots+\dfrac{1}{n\times(n+1)}=\dfrac{1}{1+\dfrac{1}{n}}$ n＝5，n＝10，n＝15のとき 右辺と左辺を別々に計算し等しくなることを確かめなさい		

練習問題 9

	計算式	キー操作	答
1	$4^3+7^2+8^4$		
2	$2.5^3+7.3^3$		
3	$4.5^{3.2}+2.5^{4.3}$		
4	$4.8^{-2}+8.9^{-3}\times 1980$		
5	$\dfrac{1}{3.8^{-1.5}+6.4^{-2.1}}$		

練習問題 10

n進数から10進数への変換

べき乗を使うとn進数から10進数への変換が簡単にできます。

たとえば　2進数101011の10進数への変換の式は、

$$(101011)_2 = 1\times2^5+\underbrace{0\times2^4}_{0}+1\times2^3+\underbrace{0\times2^2}_{0}+1\times2^1+1\times2^0 \qquad (2^0=1)$$

（各桁の指数：5 4 3 2 1 0）

$$= 1\times2^5+1\times2^3+1\times2^1+1\times2^0$$

で求めることができます。

次のn進数を10進数に変換しなさい。

	計算式	キー操作	答
1	$(11001101)_2$　2進数です		
2	$(11110000)_2$　2進数です		
3	$(123022)_4$　4進数です		
4	$(4157)_8$　8進数です		

練習問題 11

利息計算

複利による利息計算には、必ずべき乗がでてきます。そこで次の練習問題で利息計算を行い、べき乗計算に慣れてください。

	問題	計算式	答
1	10万円を年6.7%で5年間預金をするといくらになるでしょうか。	$100000\times\left(1+\frac{6.7}{100}\right)^5$　（複利計算）	
2	年利率8%のときは、月利率はいくらでしょうか。	$\left(1+\frac{8}{100}\right)^{\frac{1}{12}}-1$　（複利計算）	
3	5年間で300万円を積み立てるには、年利率7%として年にいくら掛金を積み立てたらよいでしょうか。	掛金＝元利合計×$\frac{利率}{\{(1+利率)^{期間}-1\}(1+利率)}$ $3000000\times\frac{\frac{7}{100}}{\{(1+\frac{7}{100})^5-1\}(1+\frac{7}{100})}$　（複利計算）	

練習問題 12

	計算式	キー操作	答
1	$\sqrt[4]{625}$		
2	$\sqrt[5]{1258}$		
3	$45^{\frac{5}{6}}$		
4	$23^{\frac{1}{5}}+36^{\frac{1}{4}}+43^{\frac{1}{7}}$		
5	$\sqrt[7]{12.38\times10^{20}}$		
6	$\sqrt[6]{0.4354}\times\sqrt[7]{1.875}$		

練習問題 13

	計算式	キー操作	答
1	$15!$		
2	$\frac{69!}{17!}$		
3	${}_{12}P_5$		
4	${}_{20}C_7$		
5	${}_4C_2\times{}_8C_2$		

練習問題 14

次の文を読んで計算を行い、階乗の計算に慣れましょう。

	問題文	計算式とキー操作	答
1	10人の中から４人のリレーランナーを選ぶ場合、 (a)　走る順番を決めてランナーを選ぶには (b)　走る順番を決めないでランナーを選ぶには	$_{10}P_4 =$ $_{10}C_4 =$	
2	12人の人がいます。次のような場合の数を求めなさい。 (a)　１列に並ぶ場合 (b)　円形に並ぶ場合 (c)　特定の４人が隣り合うよう12人が１列に並ぶ場合	12！ (12－１)！ ９！×４！	

練習問題 15

	計算式	キー操作	答
1	$\log 15$		
2	$\log 18.465$		
3	$\log 0.0018465$		
4	$\log 1.84\times10^{20}$		
5	$\log\sqrt[3]{\dfrac{0.95\times7.35}{0.625}}$		
6	$3.8\log 0.49$		
7	$\log\dfrac{\sqrt{81^2-3}}{15}$		
8	$\log_3 81$ ($\log_a b=\dfrac{\log b}{\log a}$ を使って)		

練習問題 16

	計算式	キー操作	答
1	$10^{3+\log 30}$		
2	$10^{-1.3}$		
3	$10^{14-13.3}$		
4	$10^{-0.012\times25+2.64}$		

練習問題 17

	計算式	キー操作	答
1	$e^{3.8}$		
2	$e^{2\times(3+7+5)}$		
3	$3.5\times e^{-\frac{2}{7}}$		
4	$\dfrac{19.3}{4.8+12.5\times e^{-3.8}}$		
5	$\dfrac{15.2}{2.3+8.5\times e^{-4}}$		

練習問題 18

	計算式		キー操作	答
1	$\cos(0.7\pi)$	(RAD)		
2	$4.5\sin^2 28°14'52''$	(DEG)		
3	$4\tan\left(\frac{\pi}{3}\right)$	(RAD)		
4	$\frac{\cos 1.286}{\tan 6.254}$	(RAD)		
5	$\sqrt{7}\cos\left(\frac{\pi+3.8}{4.6}\right)$	(RAD)		
6	$4\cos\frac{40°}{2}\times\cos\frac{60°}{2}\times\cos\frac{80°}{2}$	(DEG)		
7	$\sin 80^g+\tan 80^g$	(GRAD)		
8	$\sqrt{1-\cos^2 30°}$	(DEG)		
9	$\cos^2 30°+\sin^2 30°$	(DEG)		
10	$\frac{1}{2}\times 18\times 15.5\times\sin 30°$	(DEG)		

メモ　角度指定について

角度指定を行ってください。角度指定は 2nd F DRG で行います。

練習問題 19

	計算式		キー操作	答
1	$\sin^{-1}0.345$	(DEG)		
2	$\cos^{-1}0.345$	(RAD)		
3	$\tan^{-1}(-4.545)$	(DEG)		
4	$\sin^{-1}0.56+0.57$	(DEG)		
5	$\cos^{-1}(-0.24)-4.56$	(DEG)		
6	$\sin^{-1}0.56+0.57\times\cos^{-1}(-0.24)-4.56\times\tan^{-1}0.56$	(DEG)		
7	$\tan^{-1}\left(\frac{3\sin 60°}{4+3\cos 60°}\right)$	(DEG)		

練習問題 20

	計算式	キー操作	答
1	直交座標 (10, 10) を極座標（度）に		
2	直交座標 (32.5, −21.7) を極座標（度）に		
3	直交座標 (0.35, 0.85) を極座標（ラディアン）に		
4	極座標 $(15, \frac{\pi}{3})$ を直交座標に（πがあるからラディアンです）		
5	極座標 (73, 75°) を直交座標に		
6	極座標 $(8, -\frac{4\pi}{5})$ を直交座標に		

練習問題 21

次の統計計算を行いなさい。

問題	キー操作	答
1．学生の１カ月当りの親からもらうおこづかいを調べたところ、下表のようになりました。平均値と標準偏差（s）を求めなさい。		

お金	人数
以上　未満 0～ 500	1
500～1000	1
1000～1500	2
1500～2000	8
2000～2500	10
2500～3000	10
3000～3500	12
3500～4000	25
4000～4500	11
4500～5000	5

問題	キー操作	答
2．下表は、14人の成人男子の身長 x（cm）と体重 y（kg）を測定した結果です。相関係数を求め、体重が58kgの人の身長および身長が178cmの人の体重を推定しなさい。		

No.	身長	体重	No.	身長	体重
1	171.2	56.0	8	179.0	67.0
2	167.6	60.5	9	163.5	50.5
3	182.5	92.0	10	169.0	68.0
4	175.0	72.5	11	177.0	65.0
5	165.5	53.0	12	174.7	68.5
6	173.8	64.5	13	178.0	65.0
7	170.0	59.0	14	167.8	62.5

練習問題 22

問題	キー操作・答
1．第１図のように、糸に400ｇのおもりがつり下げられています。糸の張力T_1、T_2を求めなさい。答えは小数点以下２桁とします。	

ラミーの定理

$$\frac{W}{\sin(\alpha+\beta)}=\frac{T_1}{\sin(180^\circ-\beta)}=\frac{T_2}{\sin(180^\circ-\alpha)}$$

（ヒント）

小数点以下２桁指定はDIGIT指定で行います。

第１図

問題	キー操作・答
2．第２図のように、重さ20〔kg〕の物体を角度35°の斜面にそって５〔m〕を加速度８〔m／sec²〕で引き上げるのに４秒かかりました。この場合の仕事と仕事率を求めなさい。	

$R = W\cos\theta$

$P = W\sin\theta$

$W = 20$〔kg〕、$\theta = 35^\circ$、$S = 5$〔m〕

$a = 8$〔m／sec²〕、$t = 4$〔sec〕

$g = 9.8$〔m／sec²〕、$\mu = 0.3$

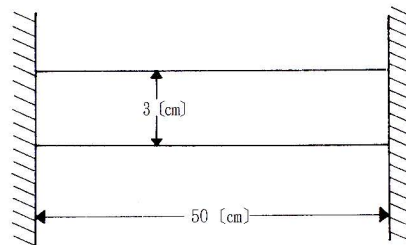
力：$F = W\left(\frac{a}{g}+\mu\cos\theta+\sin\theta\right)$

仕事：$Wp = F\cdot S$

仕事率：$Pt = F\cdot S/t$

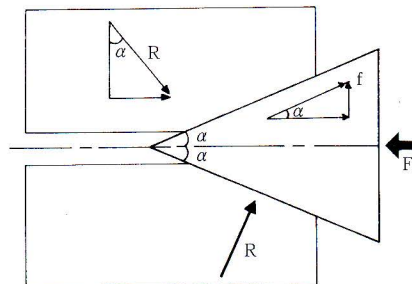
第２図

３．直径３〔cm〕、長さ50〔cm〕の銅棒の両端を固定し、棒の温度を20℃から70℃まで加熱して上昇させたとき、棒に生じる応力および、固定端の圧力を求めなさい。
ただし、
銅の線膨張係数αは　$\alpha = 0.167 \times 10^{-4}$〔$℃^{-1}$〕
銅の縦弾係数Ｅは　$E = 1.1 \times 10^{6}$〔$kg／cm^{2}$〕
とします。
熱応力：$\delta c = E \cdot \alpha (t' - t)$
圧　力：$Fc = A \cdot E \cdot \alpha (t' - t)$　（$\because A = \pi r^{2}$）

第３図

４．頂角28°の直角形くさびを木材に打ち込むのに100kg重の力を要しました。木材を引きさく力を求めなさい。ただし$\mu = 0.25$とします。
$2\alpha = 28°$
$F = 100$〔kg重〕
$\mu = 0.25$

摩擦角：$\phi = \tan^{-1} \mu$

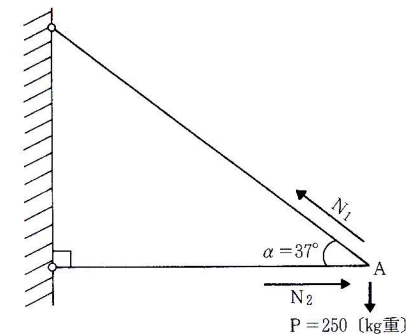
引きさく力：$R = \dfrac{F \cdot \cos \phi}{2 \sin(\alpha + \phi)}$

第４図

５．Ａ点に下向きの力Ｐがかかるとき、N_1、N_2がどのような力であればつりあいますか。

$\Sigma X = 0$（Ｘ方向の分力の和が０）
$\Sigma Y = 0$（Ｙ方向の分力の和が０）

$\Sigma X = 0$より　$-N_1 \cos \alpha + N_2 = 0$
$\Sigma Y = 0$より　$N_1 \sin \alpha - P = 0$

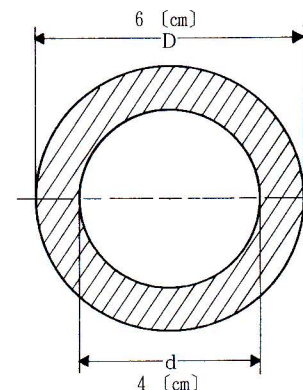
すなわち$N_1 = \dfrac{P}{\sin \alpha}$
$N_2 = N_1 \cos \alpha$

第５図

６．第６図のような外径Ｄ、内径ｄをもつ断面について、断面積Ａ、重心軸に対する２次モーメントＩ、および断面係数Ｚを求めなさい。

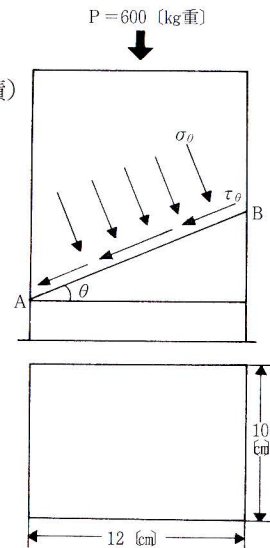
断面積　：$A = \dfrac{\pi}{4}(D^2 - d^2)$　〔cm^2〕

断面２次モーメント：$I = \dfrac{\pi}{64}(D^4 - d^4)$　〔cm^4〕

断面係数　：$Z = \dfrac{\pi}{32} \cdot \dfrac{D^3 - d^3}{D}$　〔cm^2〕

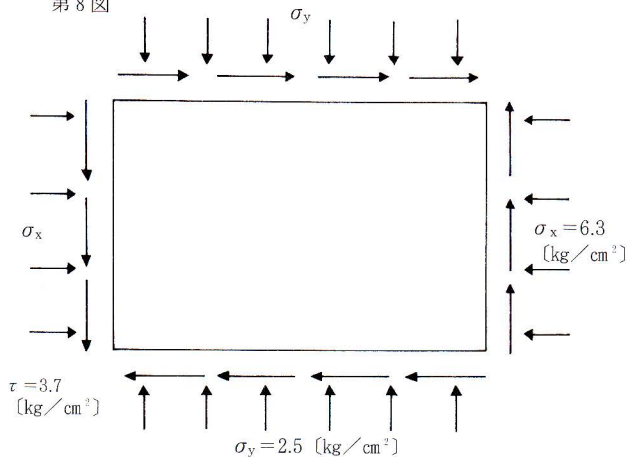
第６図

7．第7図のような場合で、θが15°、30°、45°、60°のとき、AB面に生じる垂直応力度σ_θ、せん断応力度τ_θを求めなさい。

$$\sigma = \frac{P}{A}\ 〔kg重／cm^2〕（A=面積）$$
$$\sigma_\theta = \sigma \cos^2\theta\ 〔kg重／cm^2〕$$
$$\tau_\theta = \frac{\sigma}{2}\sin 2\theta\ 〔kg重／cm^2〕$$

第7図

8．構造物内部の微小部分が第8図のような応力度を受ける場合、主応力度およびその方向を求めなさい。

主応力度の角　$\tan 2\theta = \frac{2\tau}{\sigma_x - \sigma_y}$

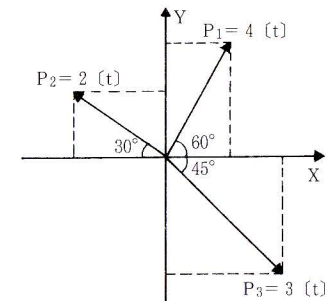
主応力度　$\sigma_{\mathrm{I.II}} = \frac{1}{2}(\sigma_x + \sigma_y) \pm \sqrt{\frac{1}{4}(\sigma_x - \sigma_y)^2 + \tau^2}$

第8図

9．第9図のような3つの力P_1、P_2、P_3の合力を求めなさい。

合力の大きさ
$$R = \sqrt{\Sigma X^2 + \Sigma Y^2}$$

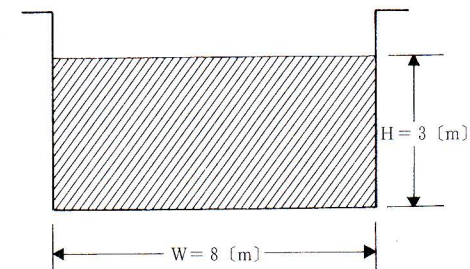
合力の向き
$$\theta = \tan^{-1}\frac{\Sigma Y}{\Sigma X}$$

$$\Sigma X = P_1\cos\theta_1 + P_2\cos\theta_2 + P_3\cos\theta_3$$
$$\Sigma Y = P_1\sin\theta_1 + P_2\sin\theta_2 + P_3\sin\theta_3$$
$$\left(\begin{array}{l}\theta_1 = 60^\circ \\ \theta_2 = 180^\circ - 30^\circ \\ \theta_3 = 360^\circ - 45^\circ\end{array}\right)$$

第9図

10．幅8〔m〕、水深3〔m〕の長方形水路の動水こう配が$\frac{1}{1500}$のとき、平均流速、流量を求めなさい。ただしマニングの粗度係数をn＝0.02とします。

流　積　$A = W \cdot H$〔m²〕
潤　辺　$S = W + 2H$〔m〕
平均流速　$V = \frac{1}{n}\left(\frac{A}{S}\right)^{\frac{2}{3}}\sqrt{I}$〔m／sec〕
流　量　$Q = V \cdot A$〔m³／sec〕

$$\left(\begin{array}{l} I = \frac{1}{1500} \\ n = 0.02\end{array}\right)$$

第10図

11. 第11図のように掘削した斜面の下に、軟弱な土質があることがわかっています。軟弱な土質の内部
摩擦力 $\phi' = 5°$
粘着力 $C' = 2$〔t／m²〕
としてこの斜面のブロックすべりに対する安全率を求めなさい。

斜面の土の内部摩擦角　$\phi = 20°$
斜面の土の粘着力　$C = 0$
斜面の土の単位体積重量　$\gamma = 1.8$〔t／m³〕とします。

● 左向きにすべらせようとして働く主働土圧：P_A

$$P_A = \frac{\gamma H_1^2}{2} \tan^2(45° - \frac{\phi}{2})$$　〔t／m〕

● すべりを止めようとして右向きに働く受働土圧：P_P

$$P_P = \frac{\gamma H_2^2}{2} \tan^2(45° + \frac{\phi}{2})$$　〔t／m〕

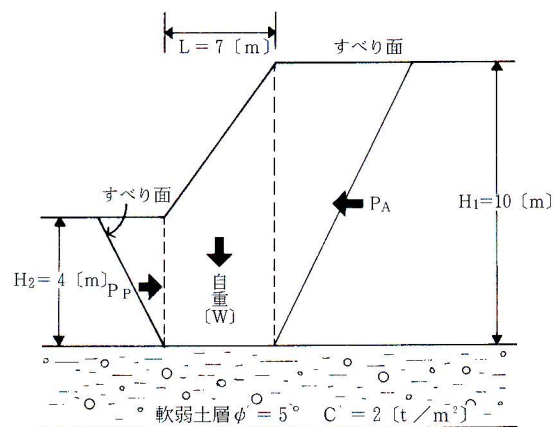
● 粘着力による抵抗：$C'L$　〔t／m〕

● 摩擦力による抵抗：$W\tan\phi = \frac{H_1 + H_2}{2} L \cdot \tan\phi'$〔t／m〕

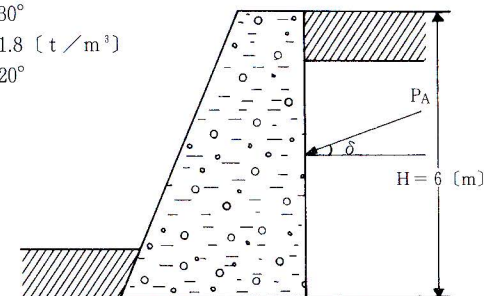
● 斜面のブロックすべりに対する安全率：F_S

$$F_S = \frac{C'L + W\tan\phi + P_P}{P_A}$$

$$F_S = \frac{C'L + \frac{H_1+H_2}{2} L\tan\phi' + \frac{\gamma H_2^2}{2}\tan^2(45° + \frac{\phi}{2})}{\frac{\gamma H_1^2}{2}\tan^2(45° - \frac{\phi}{2})}$$

第11図

12. 背部の地表面が水平な内部摩擦角 $\phi = 30°$ の土砂からなるがけに、高さ6メートルの第12図のような壁面（土と壁の摩擦角 $\delta = 20°$）を有する重力式擁壁を作りたい。
土の単位体積重量 $\gamma = 1.8$〔t／m³〕として主働土圧 P_A を求めなさい。

$\theta = 90°$
$\beta = 0°$
$\phi = 30°$
$\gamma = 1.8$〔t／m³〕
$\delta = 20°$

第12図

クーロン土圧の解析公式

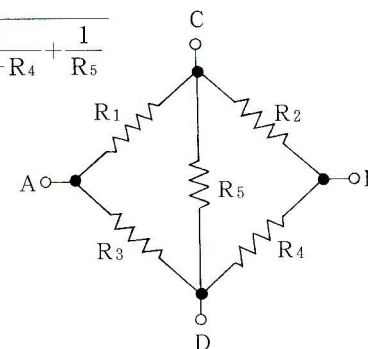
$$P_A = \frac{\gamma H^2}{2} \cdot \frac{\sin^2(\theta+\phi)}{\sin^2\theta \sin(\theta-\delta)\{1+\sqrt{\frac{\sin(\phi+\delta)\sin(\phi-\beta)}{\sin(\theta-\delta)\sin(\theta+\beta)}}\}^2}$$
〔t／m²〕

13. 第13図の回路において端子ＡＢ間、および端子ＣＤ間で測定した合成抵抗を求めなさい。

$R_1 = 400$〔Ω〕
$R_2 = 730$〔Ω〕
$R_3 = 240$〔Ω〕
$R_4 = 438$〔Ω〕
$R_5 = 600$〔Ω〕

$$R_{AB} = \frac{(R_1+R_2)(R_3+R_4)}{R_1+R_2+R_3+R_4}$$　（ブリッジが平衡しているとき）

$$R_{CD} = \frac{1}{\frac{1}{R_1+R_3} + \frac{1}{R_2+R_4} + \frac{1}{R_5}}$$

第13図

14. ある工場の三相負荷Ｐは20〔kW〕、力率60％である。これを力率80％に改善するために要するコンデンサＱ〔kVA〕を求めなさい。
また、コンデンサにかかる電圧を200〔Ｖ〕としたときの静電容量Ｃ〔μＦ〕を求めなさい。ただし、周波数は50〔Hz〕とします。

改善前の力率角 θ_1、改善後の力率角 θ_2 とするとコンデンサＱは

$$Q = P(\tan\theta_1 - \tan\theta_2)$$

静電容量は

$$3C = \frac{Q}{\omega \cdot V^2}$$

$$(\because \omega = 2\pi f)$$

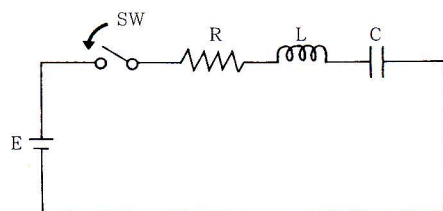
15. 第15図のようなＲＬＣ直列回路において、Ｒ＝300〔Ω〕について、スイッチＳＷを閉じてから ｔ＝５〔ms〕後の電流値を求めなさい。

● $R < 2\times\sqrt{\frac{L}{C}}$ の場合　（振動的減衰）

$$\alpha = \frac{R}{2L}$$

$$\beta = \frac{1}{2L}\sqrt{\frac{4L}{C} - R^2}$$

$$i = \frac{E}{\beta L}e^{-\alpha t}\cdot \sin\beta t$$

第15図

Ｌ＝150〔ｍＨ〕
Ｃ＝２〔μＦ〕
Ｅ＝100〔Ｖ〕

16. （ａ）試薬特級の無水炭酸ナトリウム5.2050ｇを秤量し、溶解したものをメスフラスコで１ℓに希釈し標準溶液とした。この規定度を求めなさい。

（ｂ）ある井戸水を分析したところ、100mℓ中に $CaSO_4$ 2.8mg、$Mg(HCO_3)_2$ 7.5mgが含まれていた。この水の硬度を求めなさい。

ただし、Ｈ＝1.008　Ｃ＝12.01　Ｏ＝16
Mg＝24.3　Ｓ＝32.06　Ca＝40.08
Na＝22.99
とする。

（ａ）Na_2CO_3 の１g当量＝Na_2CO_3／２である。

（ｂ）水100mℓ中の Ca 塩および Mg 塩のモル数の合計をCaOの質量ｎmgに換算し、ｎ°とする。

17. ある実験で二酸化炭素１mol が、40℃、50atm で0.380 ℓ の容積を占めることが実測された。その圧力を、
（ａ）理想気体の状態式で、
（ｂ）ファンデルワールスの状態式で
計算し、実測値と比較しなさい。

（ａ）$PV = RT \quad \rightarrow \quad P = \frac{RT}{V}$

（ｂ）$\left(P + \frac{a}{V^2}\right)(V - b) = RT \rightarrow P = \frac{RT}{V-b} - \frac{a}{V^2}$

$R = 0.0821$（ℓ・atm／（mol・K））
$a = 3.60$　　$b = 4.28\times10^{-2}$

18. 次の水溶液について質問の値を求めなさい。

（a）0.01N－HCl水溶液のpH
（b）0.05N－NaOH水溶液のpH
（c）pH＝2.50のH_2SO_4 20mℓを水でうすめて500mℓとした水溶液のpH
（d）pH＝8.50の水溶液の〔H^+〕

HCl、NaOHは完全電離とする。

$$pH = -\log\frac{Kw}{[OH^-]}$$

〔H^+〕：水素イオン濃度 (mol／ℓ)
〔OH^-〕：水酸イオン濃度 (mol／ℓ)
〔Kw〕：水のイオン積 1.0×10^{-14} (mol／ℓ)^2

19. 次の実験データがある。この分解反応が１次であることを示し、かつ、この温度における速度定数を求めなさい。ただし、初濃度a＝1.590（mol／ℓ）とします。

実験番号	1	2	3	4
時間t（min）	5.4	21.6	25.5	32.9
変化量x（mol／ℓ）	0.624	1.298	1.376	1.474

反応温度40℃

１次反応ならば $\frac{dx}{dt} = k(a-x)$

k：速度定数（min^{-1}）

t＝0のとき x＝0であるから

$k\int_0^t dt = \int_0^x \frac{dx}{a-x}$ より $k = \frac{1}{t}\ln\frac{a}{a-x}$

各実験番号におけるkを計算し、この値がほぼ一定であることを確かめてから平均値を求める。

20. 円管のまさつ係数（f）とレイノルズ数（Re）との関係式は次のようになります。

$$1/\sqrt{f} = A\log(Re\sqrt{f})+B$$

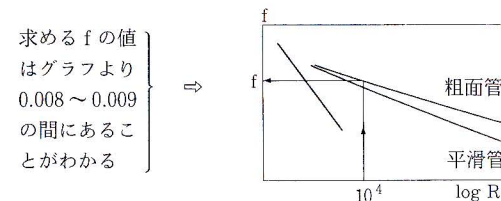
ただし、

	A	B
平滑管	4.0	－0.4
粗面管	3.2	1.2

（化工便覧）

いま、粗面管においてRe＝10000のときのfを試算法で求めなさい。

● （左辺）－（右辺）がゼロになるようにグラフで解く。

求めるfの値はグラフより0.008～0.009の間にあることがわかる ⇨

参　考

計算の優先順位

本機はカッコ、関数を含めて数式どおりのキー操作で計算を行うことができます。
計算の優先順位の判断や途中結果の処理はすべて計算機が自動的に処理してくれます。計算の優先順位は次のとおりです。

1. πや変数の呼び出し
2. 関数（SIN、COSなど）
3. べき乗（∧）
4. 符号（＋、－）
5. 乗除算（＊、／）
6. 整数の除算（¥）
7. 整数の剰余（MOD）
8. 加減算（＋、－）
9. 大小比較（＞、＞＝、＜、＜＝、＜＞、＝）
10. AND、OR、XOR

（注）● カッコが使用されている場合はカッコ内の計算が最優先されます。
● 複合関数（sin $\cos^{-1}$ 0.6など）は右から左の順で計算されます。
● べき乗の連算（3^{4^2} すなわち3∧4∧2など）は右から左の順で計算されます。
● 上記３．と４．では後に出てきたほうが優先順位が高くなります。

〈例〉 －2∧4は －(2^4) となります。
3∧－2は 3^{-2} となります。

第４章
ＢＡＳＩＣ言語

１．ＢＡＳＩＣ言語をマスターする第一歩

(1)ＢＡＳＩＣとは

Ｂ；Beginner's　（ビギナーズ）……………　初心者向きで
Ａ；All-Purpose　（オールパーパス）………　あらゆる目的に合う
Ｓ；Symbolic　（シンボリック）…………　記号を使った
Ｉ；Instruction　（インストラクション）…　命令
Ｃ；Code　（コード）……………………　語

それぞれの頭文字を取って、ＢＡＳＩＣ（ベーシック）という名のついたコンピュータ言語です。ＢＡＳＩＣはやさしい英語と記号でできており、コンピュータがどんな仕事をすればよいのかを示す一連の命令文、すなわち「プログラム」を人間とコンピュータが対話するような形で作っていけるのが大きな特長です。

(2)プログラムとは

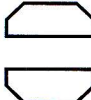
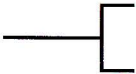
プログラムとは、計算機（コンピュータ）に計算などを行うための手順を指令する命令書のようなものです。この指令をコンピュータが理解できるように書き表したり、コンピュータに記憶させたりすることをプログラミングとか、プログラムを組むといいます。コンピュータでいろいろな問題を処理しようとするとき、最初からＢＡＳＩＣ言語でプログラミングする方法もありますが、ここではＢＡＳＩＣ言語のプログラムをマスターする第一歩として、問題を処理する方法の考えかたや、手順を整理し、図的に順序だてて、プログラムの流れをわかりやすく表現することから始めます。

(3)主な記号の読みかたと意味

記号	読みかた	意　味
＋	プラス	加　算
－	マイナス	減　算
＊	アスタリスク	乗　算
／	スラッシュ	除　算
＝	イコール	等　号
＞	グレーター・ザン	より大きい
＜	レス・ザン	より小さい

記号	読みかた	意　味
"	ダブルクォーテーション	引用符
．	ピリオド	小数点
，	コンマ	区切り
：	コロン	命令文を続けて書く
；	セミコロン	表示部をつめる
∧	べき（ハットマーク）	べき乗記号
♯	クロスハッチ	番号記号

記号	読みかた
！	イクスクラメーションマーク
？	クエッションマーク
＠	アットマーク
＆	アンドマークまたはアンパサンド
％	パーセントマーク
＄	ドルマーク
'	シングルクォーテーション

記号	読みかた
［］	大カッコ
｛｝	中カッコ
￥	円記号
¦	パイプ記号
～	波形
＿	アンダーライン

(4)「流れ図」（フローチャート　Flowchart）について

プログラムの処理方法の考えかたや手順を整理し、図的に順序だてて書いたものを「流れ図」または「フローチャート」といいます。
流れ図は、図記号と簡単な式や文字で表します。図記号はそれぞれ意味を持っています。次に代表的な図記号とその意味、そして流れ図の基本形を示しますので十分に理解してください。

(5)代表的な「流れ図」とその意味

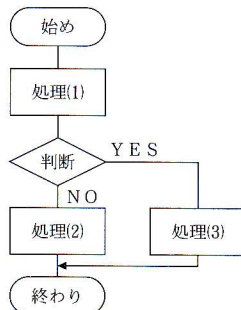
流れ図の記号	意　味
1.	**端　子**　(Terminal Interrupt) 流れ図の開始、終了などの端子を表します。
2.	**処　理**　(Process) 枠内（枠外に書いてもよい）に書かれてある処理を行います。
3.	**準　備**　(Preparation) 初期値などの準備などに用います。
4.	**入出力**　(Input／Output) 情報の入出力を意味します。入出力一般として用いられます。 また、本書では図記号 ▱ をＲＥＳＴＯＲＥとみなします。
5.	**手操作入力**　(Manual Input) 変数への入力など、キーボードなどから手で操作して入力することを表します。
6.	**定義済み処理**　(Predefined Process) サブプログラムなど、別の場所で定義されている命令群などの処理を表します。
7.	**表　示**　(Display) 情報を人間が利用できるように、ディスプレイに表示します。
8.	**判　断**　(Decision) 判断・比較を行います。

	記号	説明
9．		**書　類**　(Document) 書類を媒体とする入出力機能を表します。
10．		**繰り返しループ** ループの始まりと終わりを表します。
11．		**結 合 子**　(Connecter) 流れ図のほかの場所への出口、または、ほかの場所からの入口を表します。
12．		**流 れ 線**　(Flow Line) 記号を結びつける機能を表します。
13．		**注　釈**　(Comment Annotation) 明りょうにするために、説明または注意を加える機能を表します。

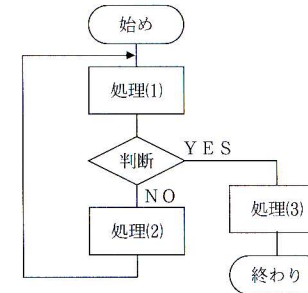
⑹プログラムの形（構造）

１．直線形の流れ図（シーケンス構造）

- 直線形プログラム
- 「判断」が入っていません。

２．分岐形の流れ図（ＩＦ～ＴＨＥＮ構造）

- 分岐形プログラム
- 設定された条件によってプログラムの流れが分岐します。
- もし（IF）、条件が満たされたら（THEN）処理(3)を行い、そうでなければ、処理(2)を行います。

３．ループ形の流れ図（ループ構造）

〈その１〉

- 繰り返し形プログラム　〈その１〉
- 条件が満たされるまで繰り返し、処理(1)と処理(2)を行います。条件が満たされると、ループを抜けて処理(3)を行います。

〈その２〉

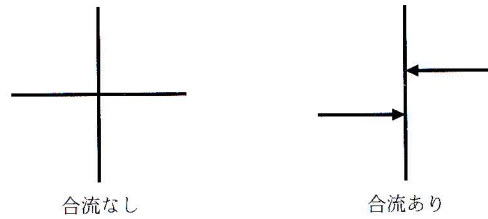
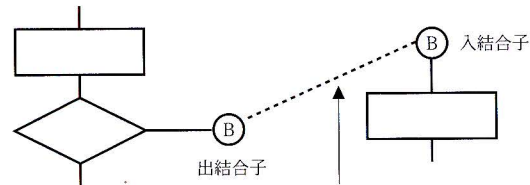
- 繰り返し形プログラム　〈その２〉
- 条件が満たされている間は、処理(1)を繰り返します。
- 条件が満たされなくなると、ループを抜けて処理(2)を行います。

⑺「流れ図」全体の約束について

①流れ図における流れ線の方向は、原則として、

上から下へ　左から右へです。

流れの方向がこれに合わないときは、流れを示す矢印を用います。

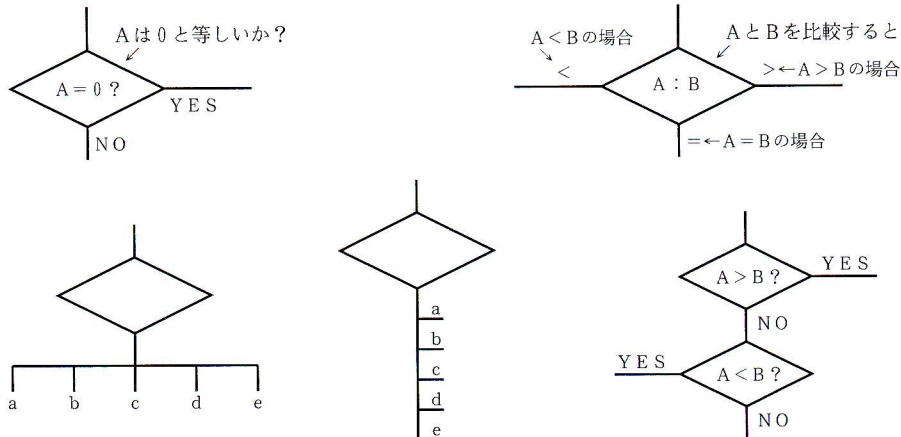
②流れ線は、交差してもかまいません。交差しても論理的関係はありません。

③結合子は、流れ線が中断される点を表すのに用いる出結合子と、中断された流れ線が再開される点を表すために用いる入結合子があります。それぞれの結合子に同じ文字や数字などを記入し、これらが結合していることを示します。

（実際は点線など書きませんが）ⒷとⒷで結合していることを示します。

④１つの流れ図記号から出口を２つ以上書く場合は、必要な数だけの流れ線をその記号から出すか、あるいは、その記号から出た流れ線の数だけ分岐する形で表現します。

分岐の出口には、分岐条件を記入します。

「＞、＝、≠、＜」などや「ＹＥＳ」、「ＮＯ」あるいは「０、１、＋、－」などがよく用いられます。

比較記号としては「：」や「？」などが使われます。

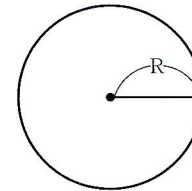
流れ図記号の中だけでなく、図記号の周辺に文章で説明する方法なども多く見受けられます。

プログラムがだんだん複雑になってくると、流れ図を書かないとプログラムの組み立てが困難になります。プログラムを組んでいるときは理解していても、しばらく時間が経過した後、あらためて解読してみるとなかなか手こずることがあります。流れ図の考えかたを早くマスターし、プログラムの基本をしっかりと身につけましょう。

2．プログラムの基本

ＳＴＥＰ 1　ＩＮＰＵＴ、ＰＲＩＮＴ、ＥＮＤ、ＧＯＴＯ文

〔例　題〕①

半径を入力して、円の面積を求めるプログラムを作りなさい。

■解　説

左図において、半径をＲ、面積をＳとすると

$S = \pi R^2$

で面積を求めることができます。

流れ図（フローチャート）	行番号（ラインナンバー）	ＢＡＳＩＣ言語によるステートメント（ひとつの意味をもった処理式や、命令語のこと）
始め		BASIC を押して画面右側に“ＰＲＯ”を点灯させます。
Ｒを入力する	10	ＩＮＰＵＴ　Ｒ
$S \leftarrow \pi R^2$	20	Ｓ＝ＰＩ＊Ｒ∧２ Ｓ＝ＰＩ＊Ｒ＊Ｒ　でもよい
面積Ｓを表示	30	ＰＲＩＮＴ　Ｓ
終わり	40	ＥＮＤ

（注）πはＰＩに変換されて入力されます。

(1)プログラムの入力

プログラムを計算機に書き込むときは、次の手順で行います。

操作の手順	操作手順の説明
① [ON] を押して電源を入れます。 [BASIC] でプログラムモード（PROモード）にします。	プログラムを書き込める状態にします。
② NEW[↵]	先に入力した（かもしれない）プログラムやデータをすべて消します。
③ 行番号を入れ、文を入れます。 ↓ 10　↓ INPUT R	[1][0][SHIFT]＋[Z](INPUT)[R] [↵]
行番号とは ──→ （行番号はラインナンバーともいいます。）	プログラムの実行順序の番号です。 数字の小さい行番号から実行されるので、実行の順に数字の小さい行番号をつけます。 行番号は、10番おきにつけるのが一般的です。後から、プログラムを追加するときに都合がよいからです。 なお、本機は1～65279までの整数を行番号として使用できます。また、行番号はAUTO（オート）命令（317ページ参照）を使うと自動発生させることができます。
表示について ──→	行番号に続けて文（ステートメント）を入れ、[↵]を押すと、行番号の後に自動的に：（コロン）が入ります。 10：INPUT␣R 自動的に1桁（1文字）あきます。 自動的に：（コロン）が入ります。
④ 20　S＝PI＊R∧2 （S＝PI＊R＊Rでも同じです。）	行番号10で与えられたRの値でπ・R^2を計算し、計算した数値をSという記号の中にしまっておきなさい（という意味です）。 [2][0][S][＝][π][＊][R][y^x∧][2][↵]
⑤ 30　PRINT　S	[3][0][SHIFT]＋[X](PRINT)[S] [↵] 行番号20のSの記号の中に入っている数値を表示しなさい（という意味です）。
⑥ 40　END ENDの意味 ──→	[4][0][E][N][D][↵] これでプログラムは終了しました（という意味です）。
[↵] について	[↵] は文の終わりに必ず入力します。 これは「この文は終わりました」または「プログラム文として書き込むことが終了しました」という意味が含まれています。

INPUTの入力方法 ──→	[SHIFT]＋[Z](INPUT) （[I][N][P][U][T]でも同じです。）
PRINTの入力方法 ──→	[SHIFT]＋[X](PRINT) （[P][R][I][N][T]でも同じです。） このように、本機ではBASIC言語の命令語が、いくつか準備されていますので、活用してください。
文（ステートメント）について 〔例題〕①をマルチ・ステートメントにすると	1つの行（ライン）は1つ以上の文（ステートメント）からなり、2つ以上の文（マルチ・ステートメント）になる場合は文と文の間に：（コロン）を入れて区別します。 （コロンを入れる） 行番号　文：文：文 [↵] 10　INPUT　R：S＝PI＊R∧2：PRINT　S：ENDとなります。ただし1行254文字以上は書けません。

NEWについて

NEW（ニュー）とは「新しい」や「初めての」という意味です。この命令は、計算機に記憶されている内容を全て消去し、初めの状態にするときに使用します。プログラムを新しく入れる前に必ず操作するBASIC命令です。

(2)プログラムの修正・編集

プログラムを作成する場合、キー操作のミスなどにより正しいプログラムにならないことが多くあります。このような場合は下表のキー操作で修正・編集します。

表示されている行（ライン）内の	[◀]または[▶]でカーソルを移動し、
① 訂　正	正しいキーの入力
② 削　除 （Delete　デリート）	[SHIFT]＋[INS](DEL)
（Backspace　バックスペース）	[BS]
③ 追　加 （Insert　インサート）	[INS]

(3)プログラムの確認

プログラム全部の内容を計算機に書き込んだら、次にPROモードで下表のキー操作により、その内容の確認をします。[CLS]を押すとプロンプト記号（>）だけの表示になります。

キ　ー　操　作	説　　　明
[▼]	・プロンプト記号（＞）が表示されているときに押すと、先頭のラインから、画面に表示できる範囲を表示します。 ・１回押すと、現在表示されている次のラインを表示します。押し続けると、順次、次のラインを表示します。 ・画面にカーソルが出ているときは、カーソルを次の行へ移します。
[▲]	・プロンプト記号（＞）が表示されているときに押すと、最終のラインを表示します。 ・１回押すと、現在表示されている前のラインを表示します。押し続けると、順次、前のラインを表示します。 ・画面にカーソルが出ているときは、カーソルを前の行へ移します。
ＬＩＳＴ[↵] またはＬ．[↵]	先頭のラインから、画面に表示できる範囲を表示します。
ＬＩＳＴ 30[↵] または Ｌ．30[↵]	行番号（ラインナンバー）30行から、画面に表示できる範囲を表示します。（指定した行番号および、それより大きい行番号が存在しないときはエラー40になります）
プログラムを呼び出した後、[▶]または[◀]を押せば表示部の１行目に表示されているプログラムラインにカーソルが表示されます。[▶]を押し続ければそのプログラムラインの最後までカーソルが移動します。[◀]の場合は先頭まで戻ります。	

(4)行（ライン）の追加・変更・削除

下の表の右側のような誤りがあったとします。

正しいプログラム	誤って入力したプログラム
10　ＩＮＰＵＴ　Ｒ 20　Ｓ＝　ＰＩ　＊Ｒ∧２ 30　ＰＲＩＮＴ　Ｓ 40　ＥＮＤ	10　ＩＮＰＵＴ　Ｒ 　＜ここに　Ｓ＝ＰＩ＊Ｒ∧２がない 20　ＰＲＩＮＴ　Ｓ 30　ＥＮＤ

誤って入力したプログラムの行番号10行と20行の間にＳ＝ＰＩ＊Ｒ∧２を入れるために、新しく15という行番号を選びます。
15　Ｓ＝ＰＩ＊Ｒ∧２[↵]とすると、下表のように訂正されたプログラムができあがります。

プログラムの内容は同じです。行番号が異なっていても問題はありません。

これで正しいプログラムが完成しました。

行番号のつけ直し

訂正されたプログラムでは、行番号が10番ごとにはなっていません。
行番号のつけ直しはＲＥＮＵＭ（リナンバー）命令（356ページ参照）を使うと簡単に行えます。

①　プログラムモードで、ＲＥＮＵＭ[↵]と操作します。
②　[▼]を押すとプログラムリストが表示されます。

```
10　INPUT　R               10　INPUT　R
15　S=　PI　*R∧2    →     20　S=　PI　*R∧2
20　PRINT　S               30　PRINT　S
30　END                    40　END
```

追加・変更・削除を行うときは、145ページの「スクリーンエディタについて」も参照してください。

(5)プログラムの実行

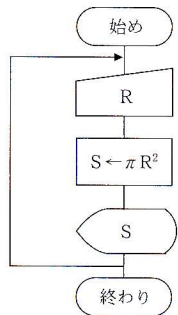
プログラムの実行は、実行モード（ＲＵＮモード）で行いますので[BASIC]を押して"ＲＵＮ"を表示させてください。

そして次の命令で実行を開始します。

ＲＵＮ[↵]　　最も小さい数字の行番号より実行を開始します。
ＲＵＮ　行番号[↵]　　指定した行番号より実行を開始します。

それでは円の面積を求めるプログラムを実行させてみましょう。

実行の内容	キー操作	表示部
ＲＵＮモードに設定		＞
実　行	[SHIFT] ＋ [V](RUN) [↵] または [R] [U] [N] [↵]	？
半径Ｒ＝１〔cm〕 の場合	１ [↵]	１＿ ３．１４１５９２６５４ ＞
半径Ｒ＝２〔cm〕 の場合	[SHIFT] ＋ [V](RUN) [↵] ２ [↵]	？ ２＿ １２．５６６３７０６１ ＞
半径Ｒ＝３〔cm〕、半径Ｒ＝４〔cm〕の場合も計算してください。		

このようにしていくと、実行のたびに [SHIFT] ＋ [V](RUN) [↵] と入力するのが、わずらわしくなってきます。
そこで下の流れ図のように、Ｓを表示した後に再び先頭に戻る命令語があれば、入力が簡単になります。
ＢＡＳＩＣ言語では、このようなときにＧＯＴＯ文を用います。

```
10  INPUT  R
20  S=  PI  *R∧2
30  PRINT  S
35  GOTO  10
40  END
```

30行と40行の間に新しくＧＯＴＯ文を入れるために
35　ＧＯＴＯ　１０ [↵] と操作します。

このプログラムを実行すると、簡単に計算を進めることができます。

実行の内容	キー操作	表示部
ＲＵＮモード	ＲＵＮ [↵]	？
Ｒ＝１の場合 次のＲの入力待ち ──→	１ [↵]	１＿ ３．１４１５９２６５４ ？
Ｒ＝２の場合 次のＲの入力待ち ──→	２ [↵]	２＿ １２．５６６３７０６１ ？

このプログラムの実行を中止するときは [ON](BREAK) を押してください。

(6)プログラムのアプリケーション

今までのプログラムの実行は入力・出力にただ数値が入っているだけで、ちょっと目を離したりすれば、それがどのような意味をもつ数値かわからなくなります。
ここではＩＮＰＵＴ文とＰＲＩＮＴ文に変化をもたせ、わかりやすい工夫を試みることにします。

※"ハンケイ␣Ｒ＝"のように " " 引用符（ダブルクォーテーション）で囲まれたものは、数値と異なり文字として扱われます。

```
30  PRINT  S
30  PRINT  "R=";R;"␣␣S=";S
```

スペースを入れないと、Ｒの数値とＳの文字がくっつきすぎてわかりにくくなります。スペースを２つ入れて、Ｒの数値とＳの文字を離しておきます。

メモ　メッセージの表示について

" " で囲まれた中に入れたアルファベットの小文字は、そのまま小文字として扱われます。
" " で囲まれた中以外でアルファベットの小文字を用いると、大文字に変換されます。ただし、注釈文（355ページＲＥＭ命令参照）は除く。

プログラムを整理すると下のようになります。

```
10  INPUT  "ハンケイ␣R=";R
20  S=  PI  *R∧2
30  PRINT  "R=";R;"␣␣S=";S
35  GOTO  10
40  END
```

これを実行してみましょう。

実行の内容	キー操作	表示部
ＲＵＮモード	ＲＵＮ ↵	ハンケイ　Ｒ＝_
Ｒ＝１ 次のＲの入力待ち ⟶	１ ↵	ハンケイ　Ｒ＝１_ Ｒ＝１．　Ｓ＝３．１４１５９２６５４ ハンケイ　Ｒ＝_
Ｒ＝２ 次のＲの入力待ち ⟶	２ ↵	ハンケイ　Ｒ＝２_ Ｒ＝２．　Ｓ＝１２．５６６３７０６１ ハンケイ　Ｒ＝_

(7)エラー表示とその処理方法

プログラムを実行したときにプログラムに誤りがあったり、データが不適当な場合などでエラーが発生することがあります。

エラーが発生すると次のようなエラーメッセージが表示されます。

```
ERROR 10 IN 20
```

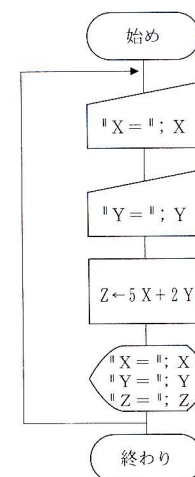
これは、行番号20行にエラーコード10の内容のエラーが発生したことを意味します。（くわしくは386ページのエラーコード表を参照）

● **エラーの処理手順**

① CLS でエラーを解除します。
② BASIC でＰＲＯモードにします。
③ ▼ または ▲ を押します。
④ エラーの発生した行番号が表示され、エラーの発生した位置にカーソルが表示されます。
⑤ エラーの発生した原因を探して、プログラムを修正します。

〔例　題〕②

ある適当な２つの数、ＸとＹを入力して、５Ｘ＋２Ｙを求めるプログラムを作りなさい。

■**フローチャート**

プログラムを入力する前に BASIC でＰＲＯモードにし、ＮＥＷ ↵ と操作します。

■**プログラム例**

```
10 INPUT"X=";X
20 INPUT"Y=";Y
30 Z=5*X+2*Y
40 PRINT"X=";X;"Y=";Y;"Z=";Z
50 GOTO 10
60 END
```

メモリ内容	
変数	内容
Ｘ	Ｘの値
Ｙ	Ｙの値
Ｚ	Ｚ＝５Ｘ＋２Ｙ

■**数値代入例**

Ｘの値	Ｙの値	Ｚの値（答）
1	2	9
5	－３	19
10	50	150
22	36	182

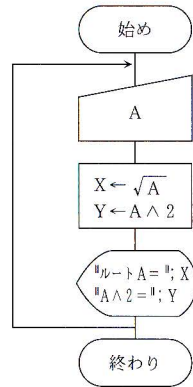
演 習 問 題　1　を行ってください。

ＳＴＥＰ ２ 切り捨て・四捨五入・桁指定

プログラムを実行し数値を求める場合、小数点以下２桁程度まで求めればよい場合が多くあります。このようなとき、切り捨て・四捨五入あるいは桁指定をして必要な桁数だけ表示するプログラムについて次の例題で練習してみましょう。

〔例　題〕③

数Ａを入力して $\sqrt{A}$, A^2 の値を表示するプログラムを作りなさい。
数Ａは１、２、３、４、5.23 を入力するものとします。

■フローチャート

■プログラム例

```
10　INPUT　A
20　X＝SQR　A
30　Y＝A∧2
40　PRINT"ルートA＝";X;"␣A∧2＝";Y
45　GOTO　10
50　END
```

■キー操作手順

操作		表示例
(BASIC)	RUNモードにします	
RUN	⏎	?
1	⏎ $\sqrt{1}$ と 1^2 の答 →	ルートA＝1.　A∧2＝1. ?
2	⏎ $\sqrt{2}$ と 2^2 の答 →	ルートA＝1.414213562　A∧2＝4. ?
3	⏎ $\sqrt{3}$ と 3^2 の答 →	ルートA＝1.732050808　A∧2＝9. ?
4	⏎ $\sqrt{4}$ と 4^2 の答 →	ルートA＝2.　A∧2＝16. ?
5.23	⏎ $\sqrt{5.23}$ と 5.23^2 の答 →	ルートA＝2.286919325　A∧2＝27. 3529 ? 表示桁数が多くなり、A^2 の値が2行にわかれてしまいました。

この〔例題〕③では、

```
40　PRINT"ルートA＝";X;"␣A∧2＝";Y
```

というように、セミコロンによって表示を継続させたので、1行に全部表示されなくなります。

しかし、次のように2つの行に分けることによって、それぞれの答えを1行に表示させることができます。

```
40　PRINT"ルートA＝";X
43　PRINT"A∧2＝";Y
```

小数点以下2桁程度を表示すればよい場合には、プログラムを次のように変更することによって $\sqrt{A}$、A^2 を一度に表示できます。

①小数点第3位以下を切り捨てる場合　ＩＮＴ／整数化（Integer／インテジャー）

```
10　INPUT　A
20　X＝SQR　A
25　X＝INT（100*X）／100　　←
30　Y＝A∧2
40　PRINT"ルートA＝";X;"␣A∧2＝";Y
45　GOTO　10
50　END
```

（Xを100倍にしたものを整数化し、それを100で割ることを意味しています。Yも同様な処理をすることができます。）

・どのような表示になるか試してみてください。

②小数点第3位以下を四捨五入する場合

```
10　INPUT　A
20　X＝SQR　A
25　X＝INT（100*X＋0.5）／100　　←
30　Y＝A∧2
40　PRINT"ルートA＝";X;"␣A∧2＝";Y
45　GOTO　10
50　END
```

（Xを100倍したものに0.5を加え、それを整数化し100で割ることを意味します。Yも同様な処理をすることができます。）

［補足］

25行はX＝（（100*X）¥1）／100としても同じです。

¥（整数の除算）については26ページを参照してください。

・どのような表示になるか試してみてください。

③数値の桁指定をする場合　ＵＳＩＮＧ命令

ＵＳＩＮＧ命令を用いて桁数を指定し表示させます。〔例題〕③で、Ｘの値を小数点2桁まで指定するときは、

```
PRINT　USING"##.##"
```

符号として1桁必要（先頭の#）　小数部は2桁分（小数点以下の##）

とします。小数点第3位以下は切り捨てとなります。

Ｙの値は符号の1桁と数値の2桁で、合わせて3桁分の指定が必要です。

```
10　INPUT　A
20　X＝SQR　A
30　Y＝A∧2
40　PRINT　USING"##.##";"ルートA＝";X
43　PRINT　USING"###";"A∧2＝";Y
45　GOTO　10
50　END
```

操　　　作		表　　示　　部
BASIC	ＲＵＮモードにします	
ＲＵＮ	↵	?
1	↵	ルートＡ＝　1．0 0 Ａ∧２＝　　1 ?
2	↵	ルートＡ＝　1．4 1 Ａ∧２＝　　4 ?
3	↵	ルートＡ＝　1．7 3 Ａ∧２＝　　9 ?
4	↵	ルートＡ＝　2．0 0 Ａ∧２＝　1 6 ?
5．2 3	↵	ルートＡ＝　2．2 8 Ａ∧２＝　2 7

四捨五入について

ＵＳＩＮＧ命令を用いたときでも、少し工夫すると四捨五入した値を表示できます。
正のときは0.005をたし、負のときは0.005を引いてから小数点第３位以下を切り捨てて表示すればよいのです。

つまり　1．234＋0．005＝1．239 ──切り捨て→ 1．23　　−1．234−0．005＝−1．239 ──切り捨て→ −1．23
　　　　1．236＋0．005＝1．241 ────→ 1．24　　−1．236−0．005＝−1．241 ────→ −1．24

負のときは、−（1．234＋0．005）のようにマイナスでくくると、正のときと同じ形になります。
正か負の判断のためにＳＧＮ（ＳＴＥＰ3 参照）という便利な関数があります。そのため求められたＸに対し、ＳＧＮ Ｘ＊（ＡＢＳ　Ｘ＋0．005）を行います。
　└ 負のときは正に符号を反転し、0．005をたします。
　└ 正のときは１、負のときは−1、ゼロのときは０になる関数です。

ＳＴＥＰ 3 関数を使うプログラム

〔例　題〕④

数Ｘを入力してその絶対値を表示するプログラムを作りなさい。

■解　　説

ＡＢＳ（Ｘ）の式でＸの絶対値を求めます。〔Absolute（アブソリュート：絶対値）〕

■フローチャート

始め → Ｘ → Ｘ←ＡＢＳ(Ｘ) → "ＡＢＳ(Ｘ)＝";Ｘ → 終わり

■プログラム例

```
10　ＩＮＰＵＴ　Ｘ
20　Ｘ＝ＡＢＳ（Ｘ）
30　ＰＲＩＮＴ"ＡＢＳ（Ｘ）＝"；Ｘ
40　ＧＯＴＯ　１０
50　ＥＮＤ
```

結　果　例	
Ｘ	ＡＢＳ(Ｘ)
3	3
5	5
−2	2
−4	4
8	8

〔例　題〕⑤

数ＸとＹを入力して、Ｘ＋Ｙの値がプラスなら１を、マイナスなら−１を、０なら０を表示するプログラムを作りなさい。

■解　　説

符号はＳＧＮで調べることができます。〔Sign（シグナム）〕
ＳＧＮ（Ｘ）で、Ｘ＞０のとき１、Ｘ＝０のとき０、Ｘ＜０のとき−１が得られます。

■フローチャート

■プログラム例

```
10　ＩＮＰＵＴ　Ｘ，Ｙ
20　Ｚ＝Ｘ＋Ｙ
30　Ｓ＝ＳＧＮ（Ｚ）　　→（　）はなくてもよい
40　ＰＲＩＮＴ"Ｘ＋Ｙ＝"；Ｓ
50　ＧＯＴＯ　１０
60　ＥＮＤ
```

結　果　例		
Ｘ	Ｙ	Ｓ
3	2	1
3	−5	−1
2	−2	0

〔例　題〕⑥

１～10の乱数を作り、表示するプログラムを作りなさい。

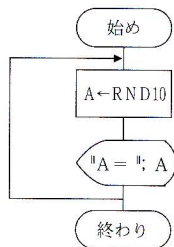
乱数はＲＮＤ　Ｘ　で与えられます。〔Random（ランダム）〕
ＲＮＤ　Ｘにおいて、Ｘの値により次のような乱数を得ることができます。

● **Ｘが２以上の場合**

Ｘが正の整数のとき：　１以上、Ｘ以下の乱数を発生します。
　　　　　　　　　　　Ｘ＝５のとき……１～５の間の乱数
Ｘが小数部を含むとき：１以上、Ｘの整数部に１を加えた値以下の乱数を発生します。
　　　　　　　　　　　Ｘ＝5．2のとき……１～６の間の乱数

● Xが負数の場合　　同じ乱数（あるいは乱数列）を発生させるために、初期値を一定にします。

● Xが0以上、2未満の場合　0より大きく、1より小さい値の乱数を発生します。
X＝1.42のとき……0～1の間の乱数

■フローチャート

■プログラム例

```
5  WAIT 50
10 A=RND 10   ←------- この部分の数値を
20 PRINT "A=";A
30 GOTO 10
40 END
```

2.8 → 小数部あり
−5 → 負　数
0.5 → 1未満

と変えて、乱数の出かたをいろいろ確認してください。

5 WAIT 50　は20行のPRINT命令の表示時間を指定しています。
ただし、一般のパソコンではWAIT指定ができないので次のようにします。

```
25 FOR J =1 TO 500:NEXT J
```

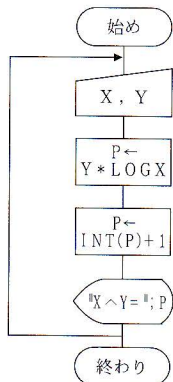
（FOR、NEXTについては121ページ参照）

〔例　題〕⑦

数XとYを入力してX^Yは何桁の数かを求めるプログラムを作りなさい。
ただし、XとYは正の整数値とします。

■解　　説

常用対数を用います。$P=X^Y$とおき、両辺の対数をとると、$\log P = Y \cdot \log X$になります。
対数の性質から、桁数はY・LOGXの値の整数値＋1桁になります。

■フローチャート

■プログラム例

```
10 INPUT X,Y
20 P=Y*LOG X
30 P=INT (P)+1
40 PRINT X;"∧";Y;"=";P;"ケタ"
50 GOTO 10
60 END
```

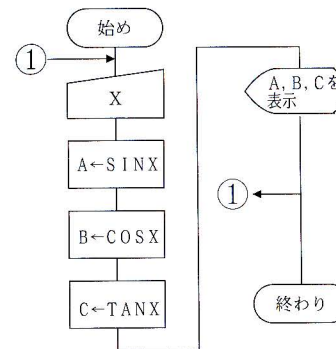
メモリ内容	
変　数	内　容
X	入力値
Y	入力値
P	桁　数

〔例　題〕⑧

θに適当な数を入れ sin，cos，tan の値を求めるプログラムを作りなさい。
（θ＝30, 45, 60と入力しなさい。）

■解　　説

三角関数キーを用います。角度単位を度（DEG）に指定します。

■フローチャート

■プログラム例

```
5   DEGREE
10  INPUT X
20  USING
30  A=SIN X:B=COS X:C=TAN X
40  PRINT "SIN";X;"="
     ;USING"##.####";A
50  USING
60  PRINT "COS";X;"="
     ;USING"##.####";B
70  USING
80  PRINT "TAN";X;"="
     ;USING"##.####";C
90  GOTO 10
100 END
```

〰〰 線で示す内容がある場合と、ない場合の表示のしかたの違いを確認してください。

メモリ内容	
変　数	内　容
X	入力値
A	SIN X
B	COS X
C	TAN X

メモ　USINGについて

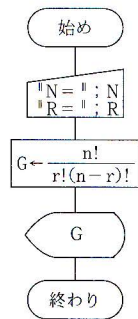
〔例題〕⑧の20行を20 USING"###"と変更し、実行してみてください。
「SIN 30＝」と30に小数点がつかない形で表示されます。（33ページ参照）

〔例　題〕⑨

組合せ計算 $nCr=\frac{n!}{r!(n-r)!}$ の値を求めるプログラムを作りなさい。

■解　　説

組合せ（NCR）を用いれば簡単に計算できますが、ここでは階乗（FACT）を用いてプログラムを作ってみます。

■フローチャート

■プログラム例

```
10  INPUT "N＝"；N，"R＝"；R
20  G＝FACT N／(FACT R＊FACT (N−R))
30  PRINT G
40  END
```

（注）20行はＧ＝ＮＣＲ（Ｎ，Ｒ）でもよい。

メモリ内容	
変　数	内　容
N	入力値
R	入力値
G	n!／r!(n−r)!

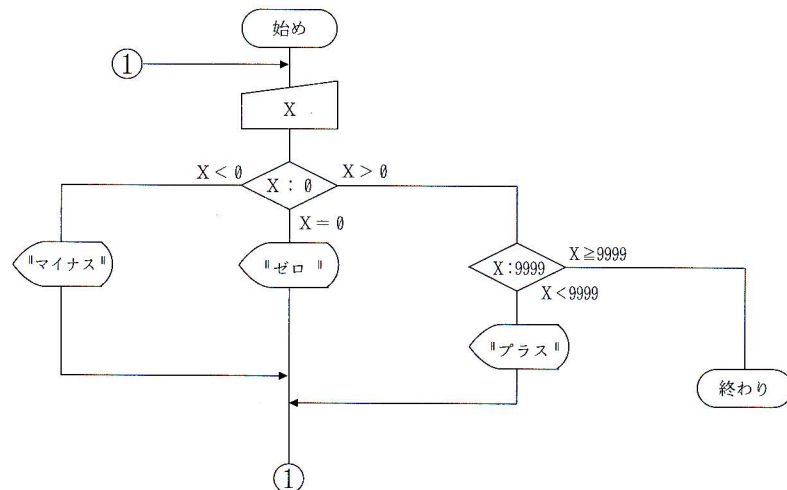
結果例		
n	r	nCr
5	3	10
10	4	210

演習問題　2　を行ってください。

ＳＴＥＰ ４　ＩＦ～ＴＨＥＮ～ＥＬＳＥ

〔例　題〕⑩

数Ｘを入力し、正の数なら"プラス　デス"
　　負の数なら"マイナス　デス"
　　０なら"ゼロ　デス"
と表示するプログラムを作りなさい。ただし、データが9999以上のときは終わるプログラムを作りなさい。

■フローチャート

■プログラム例

```
10  INPUT "X＝"；X
20  IF X＜0 THEN 70
30  IF X＝0 THEN 90
40  IF X＞＝9999 THEN 110
50  PRINT "プラス デス"
60  GOTO 10
70  PRINT "マイナス デス"
80  GOTO 10
90  PRINT "ゼロ デス"
100 GOTO 10
110 END
```

（20行）もし、Ｘの値が負なら70行へ行きなさい。そうでないなら次の行に進みます。
（30行）Ｘの値が０なら90行へ行きなさい。そうでないなら次の行に進みます。
（40行）Ｘの値が9999以上なら110行へ行きなさい。そうでないなら次の行に進みます。

（注）ＥＬＳＥを省略した例です。

■解　説

① ＩＦ～ＴＨＥＮ～ＥＬＳＥは、ＩＦ以降の条件が成立するときはＴＨＥＮ以降の命令に従い、条件が成立しないときはＥＬＳＥ以降の命令に従います。ＥＬＳＥを省略した場合は、条件が成立しないとき、次の行に進みます。
ＴＨＥＮ（またはＥＬＳＥ）　行番号 ……………… その行番号へ行きます。
ＴＨＥＮ（またはＥＬＳＥ）　ステートメント ……… そのステートメントを実行した後、次の行番号へ行きます。

② ＩＦ～ＴＨＥＮ　行番号はＩＦ～ＧＯＴＯ　行番号と同じ命令です。
「ＴＨＥＮ ＧＯＴＯ 行番号」という命令文は、「ＴＨＥＮ 行番号」または「ＧＯＴＯ 行番号」という形に省略できます。
「ＥＬＳＥ ＧＯＴＯ 行番号」は、「ＥＬＳＥ 行番号」という形に省略できます。（「ＧＯＴＯ 行番号」にはできません。）
〈例〉１０　ＩＦ　Ａ＝５　ＴＨＥＮ　Ｂ＝０　ＥＬＳＥ　２００
　Ａが５ならばＢを０にして次の行に移り、Ａが５以外ならば２００行に移ります。

条　件　式	判　断　内　容
○○＝××	等しいかどうか判断　（○○は××に等しいか？）
○○＞××	大きいかどうか判断　（○○は××より大きいか？）
○○＞＝××	以上かどうか判断　（○○は××以上か？）
○○＜××	小さいかどうか判断　（○○は××より小さいか？）
○○＜＝××	以下かどうか判断　（○○は××以下か？）
○○＜＞××	等しくないかどうか判断（○○と××は等しくないか？）

〔例　題〕⑪

二次方程式ａｘ²＋ｂｘ＋ｃ＝０（ａ≠０）の解の種類を判別して解を求めるプログラムを作りなさい。ただし、２実根のときは"２ジッコン"、重根のときは"ジュウコン"、虚根のときは"キョコン"と表示させてそれぞれの解を求めなさい。

■解　説

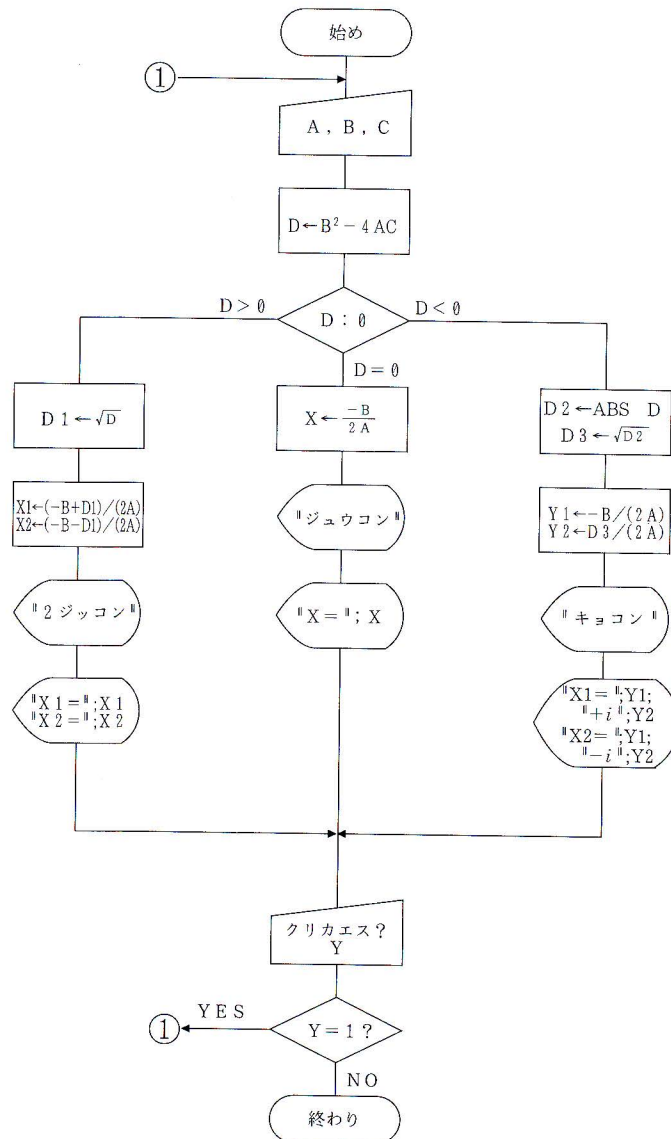
一般式　$ax^2+bx+c=0$　（$a\neq 0$）の解の公式は $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$

$D=b^2-4ac$ とすると

$D>0$ ……………２実根（$x_1=\dfrac{-b+\sqrt{D}}{2a}$，$x_2=\dfrac{-b-\sqrt{D}}{2a}$）

$D=0$ ……………重　根　$x=\dfrac{-b}{2a}$

$D<0$ ……………虚　根（$x_1=\dfrac{-b+\sqrt{D}i}{2a}$，$x_2=\dfrac{-b-\sqrt{D}i}{2a}$）

■フローチャート

メモ 虚数について

√ の中が負になると、一般にコンピュータではエラーになり計算できません。
そのため、Dの符号を判断し、Dの絶対値を考えて計算する必要があります。
数学では、$i^2=-1$ になる記号を考え出し、$\sqrt{-D}=\sqrt{D}\,i$ と表します。i を虚数単位と呼びます。
なお、電気関係では電流を i と表現しますので、虚数単位を j と表すこともあります。

■プログラム例

```
10  INPUT"A=";A,"B=";B,"C=";C
20  D=B∧2-4*A*C
30  IF D<0 THEN 160
40  IF D=0 THEN 120
50  D1=SQR D
60  X1=(-B+D1)/(2*A)
70  X2=(-B-D1)/(2*A)
80  PRINT"2ジッコン"
90  PRINT"X1=";X1
100 PRINT"X2=";X2
110 GOTO 210
120 X=-B/(2*A)
130 PRINT"ジュウコン"
140 PRINT"X=";X
150 GOTO 210
160 D2=ABS D:D3=SQR D2
170 Y1=-B/(2*A):Y2=D3/(2*A)
180 PRINT"キョコン"
190 PRINT"X1=";Y1;"+i";Y2
200 PRINT"X2=";Y1;"-i";Y2
210 INPUT"クリカエス?␣(YES=1)";Y
220 IF Y=1 THEN 10
230 END
```

メモリ内容	
変数	内容
A	2次係数
B	1次係数
C	定数
D	B^2-4AC 判別式
D1	$\sqrt{D}$
D2	ABS D
D3	$\sqrt{D2}$
X	－B／2A
X1	(－B＋D1)／2A
X2	(－B－D1)／2A
Y1	－B／2A
Y2	D3／2A
Y	繰り返し判断用

下表の数値を入れて結果を求めてください。

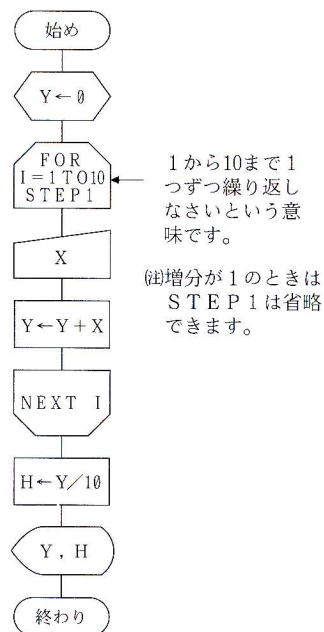
A	B	C	結果
1	5	6	2ジッコン X1=－2 X2=－3
1	2	1	ジュウコン X=－1
5	－10	25	キョコン X1=1+i2 X2=1－i2
1	1	$-3+\sqrt{3}$	2ジッコン X1=7.320508075×10^{-1} X2=－1.732050808
1	－3	6	キョコン X1=1.5+i1.936491673 X2=1.5－i1.936491673
2	$2\sqrt{10}$	5	ジュウコン X=－1.58113883

演習問題 3 を行ってください。

ＳＴＥＰ 5 ＦＯＲ～ＴＯ～ＳＴＥＰ、ＮＥＸＴ

〔例　題〕⑫

数値を10個入力して合計と平均を求めるプログラムを作りなさい。

■フローチャート

■プログラム例

```
10  Y＝0
20  FOR  I＝1  TO  10  STEP  1
30  INPUT"X＝"；X
40  Y＝Y＋X
50  NEXT  I   ←  対応しています
60  H＝Y／10
70  PRINT"ゴウケイ␣"；Y
80  PRINT"ヘイキン␣"；H
90  END
```

メモリ内容	
変数	内　容
Y	初期値設定 Y＝0 累　計 Y＋X
X	入力値
H	平　均 Y／10

■解　　説

①行番号10のＹ＝０は、変数Ｙの中には何も数値が入っていないという状態を作っています。この状態を最初に作っておかないと、行番号40のところで誤った値を加算することになります。

②行番号20のＳＴＥＰ　１はこの値が１の場合に限って省略できます。
ＦＯＲ　ＩのＩとＮＥＸＴ　ＩのＩは同じ文字（変数）にしなければなりません。
ただし、ＮＥＸＴ　ＩのＩは省略できます。たとえば、ＦＯＲ　Ｊ＝１　ＴＯ　10とすれば、ＮＥＸＴ　Ｊとするか、ＮＥＸＴということになります。
しかし、慣れるまではＮＥＸＴの後の文字を省略しないほうがよいでしょう。
プログラムを見直すときは、ＮＥＸＴの後の文字があるほうが便利です。

③行番号40のＹ＝Ｙ＋Ｘは累計を求めています。つまり、Ｙに新しいＸの値を加えています。

④行番号60のＨ＝Ｙ／１０は平均を求めています。

■累計の数値例

Ｉの値	1	2	3	4	5	6	7	8	9	10
Ｙ＋Ｘ	0＋1	1＋2	3＋3	6＋4	10＋5	15＋6	21＋7	28＋8	36＋9	45＋10
Ｙ＝Ｙ＋Ｘの値	1	3	6	10	15	21	28	36	45	55

● 上の表のようにＹの値はＩの値の変化とともにそれぞれ累計されていきます。

ＦＯＲ～ＮＥＸＴ文を使用しないで合計を求めるプログラム

〔例　題〕⑫で、ＦＯＲ～ＮＥＸＴ文を使用しないで累計を求める方法

■プログラム例

```
10   A＝0
20   Y＝0
30   INPUT"X＝"；X
40   A＝A＋1
50   Y＝Y＋X
60   IF  A＜10  THEN  30
70   H＝Y／10
80   PRINT"ゴウケイ＝"；Y
90   PRINT"ヘイキン＝"；H
100  END
```

■解　　説

①このプログラムは30行と60行の間を条件が成立するまで、繰り返しています。（繰り返しループを形成します）

②40行のＡ＝Ａ＋１は左辺のＡが10になるまで60行のＩＦ文で判断され、このループで10回のたし算を行います。
Ａをカウンターとして利用しています。

③50行のＹ＝Ｙ＋Ｘで、Ｘの値が左辺のＹの変数に累計されます。Ａが10になった時点で累計完了ということになります。

研究演習問題

データが10個ではなく、Ｎ個の場合（Ｎは任意の正の整数）の合計と平均を求めるプログラムを考えてください。

「ＦＯＲ～ＮＥＸＴ」文の二重ループ

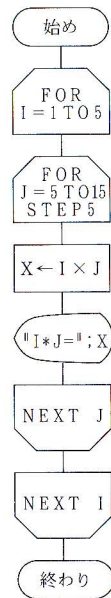
〔例　題〕⑬

1×5，1×10，1×15
2×5，2×10，2×15
⋮
5×5，5×10，5×15
をそれぞれ表示し、答えも表示するプログラムを作りなさい。

■解　　説①

１～５の１つ刻みのループ（ＳＴＥＰ　１）と
５～15の５つ刻みのループ（ＳＴＥＰ　５）
がありますので、ＦＯＲ～ＮＥＸＴの二重ループとなります。

■フローチャート

始め → FOR I＝1TO5 → FOR J＝5TO15 STEP5 → X←I×J → "I＊J＝"；X → NEXT J → NEXT I → 終わり

■プログラム例

```
10 FOR I＝1 TO 5
20 FOR J＝5 TO 15 STEP 5
30 X＝I＊J
40 PRINT I；"＊"；J；"＝"；X
45 FOR K＝1 TO 500：NEXT K
50 NEXT J
60 NEXT I
70 END
```

■解　説②

行番号20と50が対応し、行番号10と60が対応しています。ＩとＪの値について、ループ内での数値の動きは

Ｉ＝１ → Ｊ＝５，Ｊ＝１０，Ｊ＝15となってＩ＝２に移ります。
Ｉ＝２ → Ｊ＝５，Ｊ＝１０，Ｊ＝15となってＩ＝３に移ります。
Ｉ＝３ → Ｊ＝５，Ｊ＝１０，Ｊ＝15となってＩ＝４に移ります。
Ｉ＝４ → Ｊ＝５，Ｊ＝１０，Ｊ＝15となってＩ＝５に移ります。
Ｉ＝５ → Ｊ＝５，Ｊ＝１０，Ｊ＝15

となり、これでプログラムは終了となります。

■ＦＯＲ～ＮＥＸＴについて

〈例１〉このプログラム例が正しい使いかたです。ループ①はループ②の中に完全に入っていなければなりません。使用できる最大のループ数については327ページのＢＡＳＩＣの各命令の説明を参照してください。

```
10  FOR I＝1 TO 5  ───────┐
 ：                          │
40  FOR J＝1 TO 10 ──┐      │
 ：                   │ループ① │ループ②
80  NEXT J ──────────┘      │
 ：                          │
100 NEXT I ─────────────────┘
```

〈例２〉

```
10  FOR I＝1 TO 5
 ：
40  FOR J＝1 TO 10
 ：
80  NEXT I
 ：
100 NEXT J
```


このような、プログラムはERROR52となります。ループ①とループ②が交差することは文法上許されません。

〈例３〉

```
40  GOTO 80
 ：
60  FOR B＝－10 TO 12 STEP 2
 ：
80  D＝B∧2－4＊A
 ：
100 NEXT B
```

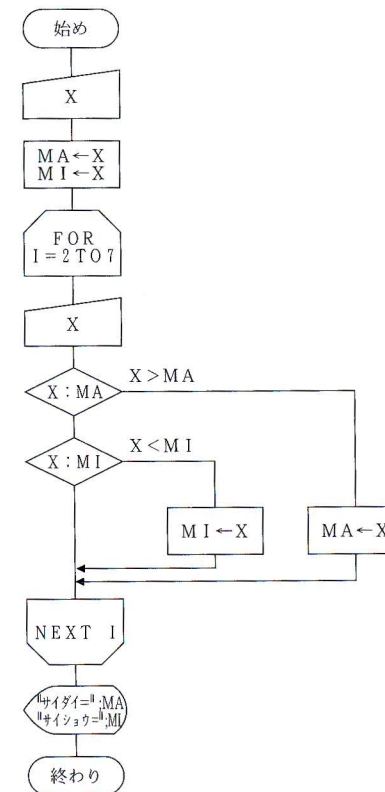

外からＦＯＲ～ＮＥＸＴループ内にとび込むことはできません。

〔例　題〕⑭

> ７つのデータがあります。15、21、36、81、９、16、10をそれぞれ入力し、最大値と最小値を求めるプログラムを作りなさい。

■フローチャート

始め → X → MA←X MI←X → FOR I＝2TO7 → X

NEXT I → "サイダイ＝"；MA "サイショウ＝"；MI → 終わり

■プログラム例

```
10 INPUT"X＝"；X
20 MA＝X：MI＝X
30 FOR I＝2 TO 7
40 INPUT"X＝"；X
50 IF X＞MA THEN MA＝X：GOTO 70
60 IF X＜MI THEN MI＝X
70 NEXT I
80 PRINT"ｻｲﾀﾞｲ＝"；MA；"ｻｲｼｮｳ＝"；MI
90 END
```

メモリ内容	
変数	内　容
MA	最大値
MI	最小値
X	データ入力値

■解　説

①行番号10で最初のデータを入力します。
②行番号20のＭＡ＝Ｘ：ＭＩ＝Ｘで、最初の入力データを仮の最大値、最小値にします。
③行番号30～70で、データを１つずつ入力していきます。
④行番号50～60で、最大値、最小値の比較を行っています。

下の表で最大値、最小値を比較している内容を細かく分析してみます。

Xの値	最大値の比較 X＞MA	最大値	最小値の比較 X＜MI	最小値
15のとき	−	15となる	−	15となる
21のとき	21＞15	21に変わる	21＜15	15のまま
36のとき	36＞21	36に変わる	36＜15	15のまま
81のとき	81＞36	81に変わる	81＜15	15のまま
9のとき	81＞9	81のまま	9＜15	9に変わる
16のとき	16＞81	81のまま	16＜9	9のまま
10のとき	10＞81	81のまま	10＜9	9のまま

↑最終結果・最大値　　↑最終結果・最小値

演 習 問 題 4　を行ってください。

STEP 6 REM、READ、DATA、RESTORE

〔例　題〕⑮

55、70、80、65、50を読んで、平均を求めるプログラムを作りなさい。ただし、ヘイキンという注釈文をプログラムリストの最初につけなさい。

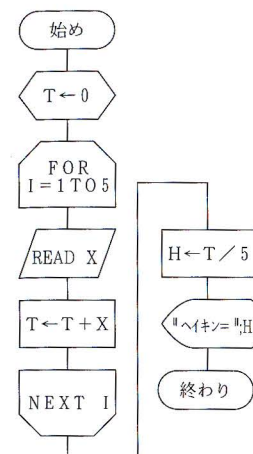
■解　説

①注釈文をつけるときは、REM文を用います。(REMまたは ’)
REMは、注釈の意味でプログラムの実行には関係ありません。しかし、プログラムリストをみるとどのようなプログラムなのかがわかります。
また、長いプログラムなどにおいては、演算処理の項目ごとにREM文で注釈をつけておくと、後でプログラムリストを検討するときにとてもわかりやすくなります。
②REM文は、任意の行番号をつけることができます。
③READ文は、DATA文と対になって変数にデータを割りあてます。
④対応するデータがなくなると、ERROR53が発生します。

■プログラム例

```
10  REM  ヘイキン
20  T=0
30  FOR  I=1  TO  5
40  READ  X
50  T=T+X:REM  ゴウケイ
60  NEXT  I
70  H=T/5
80  PRINT"ヘイキン=";H
90  END
100 DATA  55, 70, 80, 65, 50
```


■フローチャート

■結　果

ヘイキン＝64.

■プログラムの解説

10　ヘイキンという注釈文になります。10’ヘイキン　としてもかまいません。
20　Tの変数を0（ゼロ）にします。ここに、CLEAR命令を用いてもかまいません。
30　30行から60行のFOR～NEXTループです。Iの値が1から始まって、5を超えるまで繰り返します。
40　READ文です。
READ　Xは、100行のDATA文の最初のデータ55を読み込みます。
50　累計を求めています。このときのTは、T＝T＋X（右辺のT：20行の0（ゼロ）、X：DATA 55、左辺のT：55）
60　NEXT　Iで30行に戻ります。このとき、すでに、I＝2となっています。
40行のREAD　Xは、再度100行の次のDATA文の70を読み込みます。
50行は　T＝T＋X　←DATA 70（右辺のT：I＝1のときのTの合計 55、左辺のT：125）
このように、残りのデータ80、65、50も同じように読み込み、たしていきます。
70　H＝T／5（T：320が入ります（5つのデータの合計です））
したがって、変数Hには、64という数字が入ります。
80　Hの値を表示します。
90　プログラムの実行が終了します。
100　DATA文です。

★★★★★

行番号50の：REM　ゴウケイ　は、注釈文です。：(コロン）は継続行を意味しますから、このような形でREM文を使うこともできます。REM文で注釈をつけておくと、後でプログラムリストを解析するときにとてもわかりやすくなります。

〔例　題〕⑯

> 下のデータを用いて、A＝60、B＝50、C＝10、D＝40、E＝20、F＝30、G＝60、H＝50となるように、データを読み込むプログラムを作りなさい。
> 60、50、10、40、20、30

■解　　説

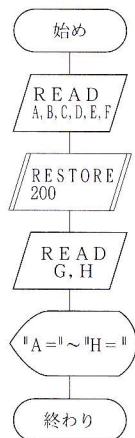
ＲＥＳＴＯＲＥ　行番号は、データを再び行番号の先頭から読み込む命令文です。
同じデータを何回も使用するときは、効果的な手段となります。

■フローチャート

■プログラム例

```
10   REM RESTORE
20   READ A, B
30   READ C, D, E, F
40   RESTORE 200
50   READ G, H
60   PRINT"A=";A;"␣B=";B
70   PRINT"C=";C;"␣D=";D
80   PRINT"E=";E;"␣F=";F
90   PRINT"G=";G;"␣H=";H
100  FOR J=1 TO 1000:NEXT J
110  END

200  DATA 60, 50
210  DATA 10, 40, 20, 30
```

200行の60：20行でAに対応、50行でGに対応
200行の50：20行でBに対応、50行でHに対応
210行：30行でそれぞれCに対応，Dに対応，Eに対応，Fに対応

■結　　果

```
A=60  B=50
C=10  D=40
E=20  F=30
G=60  H=50
```

■プログラムの解説

10　注釈文です。
20　ＲＥＡＤ　ＡのＡは200行のＤＡＴＡ文にある60を読み込み、Ｂは50を読み込みます。
30　ＲＥＡＤ　Ｃ，Ｄ，Ｅ，Ｆは、210行の、10、40、20、30をそれぞれ順次読み込んで行きます。
40　200行のＤＡＴＡ文を再び呼び出します。
50　ＲＥＡＤ　ＧのＧには200行のＤＡＴＡ文の60、Ｈには50を読み込みます。
60～110　それぞれ、読み込んだデータを表示します。110行でプログラムの実行は終わります。

なお、100行は画面をしばらく止めておくために入れています。

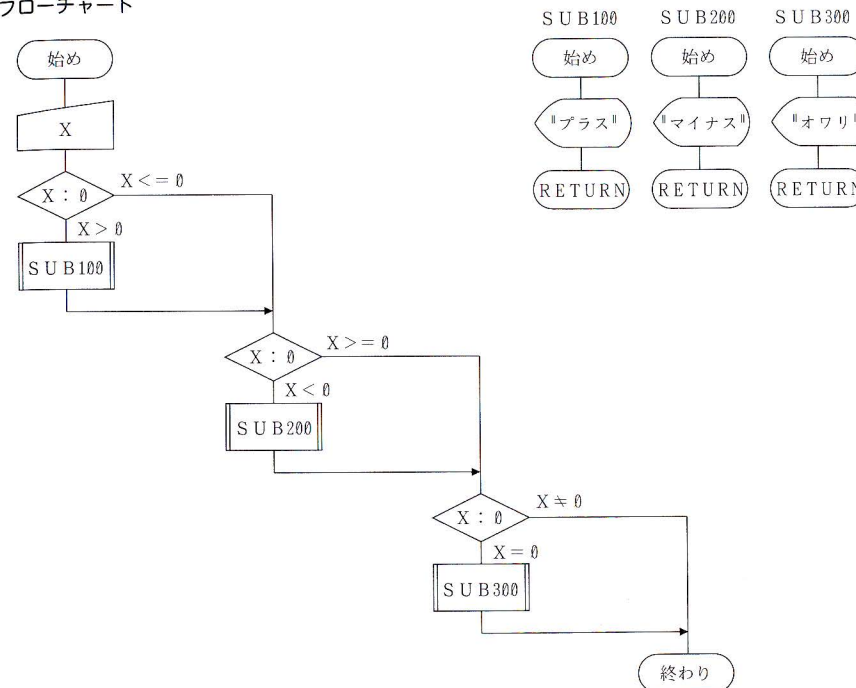
★★★　200行と210行は、ＤＡＴＡ文です。

演 習 問 題　5　を行ってください。

ＳＴＥＰ ７　ＧＯＳＵＢ～ＲＥＴＵＲＮ

〔例　題〕⑰

> 数Ｘを入力してプラスのときは"プラス"、マイナスのときは"マイナス"、０のときは"オワリ"と表示するプログラムを作りなさい。

■フローチャート

■プログラム例

```
10   INPUT X
20   IF X>0 THEN GOSUB 100
30   IF X<0 THEN GOSUB 200
40   IF X=0 THEN GOSUB 300
50   END
100  PRINT"プラス"
110  RETURN
200  PRINT"マイナス"
210  RETURN
300  PRINT"オワリ"
310  RETURN
```

■解　　説

①独立した１つのプログラムで、いつでも呼び出して使えます。

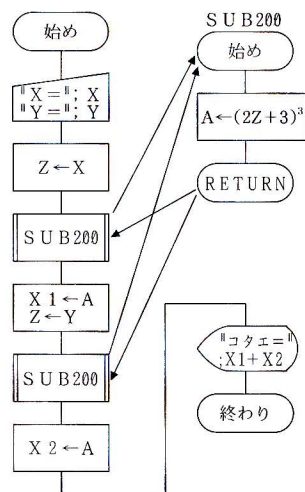
②呼び出しかたは**ＧＯＳＵＢ　行番号**となります。

③サブプログラムの終わりはＲＥＴＵＲＮです。

④このプログラムの実行順序はＸ＝－３とすれば、10→20→30→200→210→40→50です。

〔例　題〕⑱

数XとYを入力し、(2X+3)^3+(2Y+3)^{3}を求めるプログラムを作りなさい。

■フローチャート

結果例		
X	Y	コタエ
1	1	250
1	2	468
5	10	14364

■解　説

①(2X+3)^{3}と(2Y+3)^{3}はXとYをZに置き換えると同じ形となります。この計算式をサブプログラムにしておくと、何度も書く必要がなくなり簡単な形のプログラムにできます。

②サブプログラムで(2X+3)^{3}を求めるためにZ=Xとして、Xの値をZに代入しておきます。(ここが重要です)

③サブプログラムの計算結果はX1=AとしてX1に代入します。

④同じようにZ=Yとして、Yの値をZに代入しておきます。

⑤サブプログラムの計算結果はX2=AとしてX2に代入しておきます。

■プログラム例

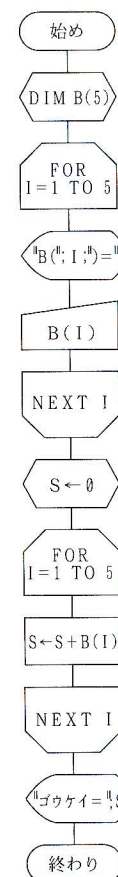
```
10　INPUT"X=";X
20　INPUT"Y=";Y
30　Z=X
40　GOSUB　200
50　X1=A
60　Z=Y
70　GOSUB　200
80　X2=A
90　PRINT"コタエ=";X1+X2
100　END
200　A=(2*Z+3)∧3
210　RETURN
```

演習問題 6　を行ってください。

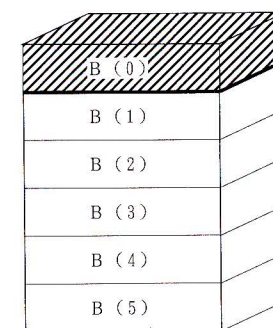
STEP 8　配列DIM(ディメンジョン)

〔例　題〕⑲

5個の数　25、18、23、17、9を読み込んで、読み込んだ順に配列変数Bに代入し、次に5個の数の合計を求めて表示するプログラムをDIM文を用いて作りなさい。

■フローチャート

1次元配列

- 添字(そえじ):これは配列の大きさです。この数字は必ず整数を用います。
- 数値配列変数　英文字を1文字または2文字使用できます。また、英文字と数字の組み合わせも使用できます。
- 配列(ディメンジョン)を意味します。変数を下図のように、データをしまい込む(出し入れをする)箱と考えたとき、DIM文は箱の数を決めて用意するステートメントと考えます。

DIM　B(5)は

- B(0)からB(5)までB(0)を含めて、6個の箱が用意されます。
- この例題のプログラム例では斜線の部分は未使用ということになります。

配列の添字について

配列の大きさを指定する数値を添字(そえじ)といいます。添字が1つのものを1次元配列(例:B(5))、添字が2つのものを2次元配列(例:B(5,2))といいます。添字がn個あれば、n次元配列ということになります。本機は2次元配列までできます。
また、コンピュータでは一般に、B(0)やB(0,0)のように0から始まる変数が確保されるので、人間の感覚とは若干ずれます。

■プログラム例

```
10  DIM  B (5)
20  FOR  I=1  TO  5
30  PRINT"B (";I;") =";
40  INPUT  B (I)
50  NEXT  I
60  S=0
70  FOR  I=1  TO  5
80  S=S+B (I)
90  NEXT  I
100 PRINT"ゴウケイ=";S
110 END
```

結　果　例
B(1)=25
B(2)=18
B(3)=23
B(4)=17
B(5)= 9
ゴウケイ=92

このようにB(I)のIには、順次1、2、3、4、5という数字が代入されていきます。

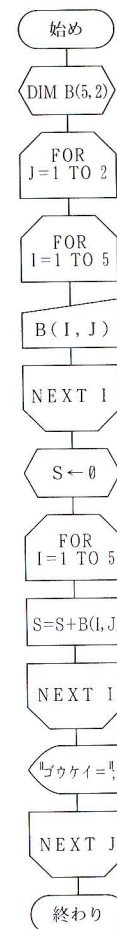
■プログラムの解説

10　配列を指定します。
20　5個（入力する値が5個）の数値が入るループを作ります。
30　入力する順番を表示させます。最後の；は、40行で入力される内容を続けて表示させるために入れています。
40　数値を入力します。30行で表示した内容の後に入ります。
50　5個の値を入力するまで20行へ行きます。
60　合計を求める前に、今の合計値を0に指定します。（まだ計算をしていないためです。）ここでCLEAR命令を使うと、せっかく入力した5個の値が消えてしまいます。
70　B(I)の箱の中に入れた5個の数値を1から5まで取り出すループを作ります。
80　1つずつ取り出したら、それぞれ累計します。
90　5個の値をB(I)から取り出すまで70行へ行きます。
100　5個の値を全て取りだした後、累計した結果を表示します。
110　プログラムの実行が終了します。

〔例　題〕⑳

右の図のように5個の数が2列あります。
列ごとの合計を求めて表示するプログラムを作りなさい。

	25	38
	18	46
	23	92
	17	73
	9	65
合　計		

■フローチャート

2次元配列
　DIM　B(5，2)
　　　　(行，列)
　　5行，2列の配列を意味します。
DIM　B(5，2)の2次元配列では下図のような箱が用意されます。

B(0,0)	B(0,1)	B(0,2)
B(1,0)	B(1,1) 25	B(1,2) 38
B(2,0)	B(2,1) 18	B(2,2) 46
B(3,0)	B(3,1) 23	B(3,2) 92
B(4,0)	B(4,1) 17	B(4,2) 73
B(5,0)	B(5,1) 9	B(5,2) 65

例題⑳では斜線の部分は使用していません。

■プログラム例

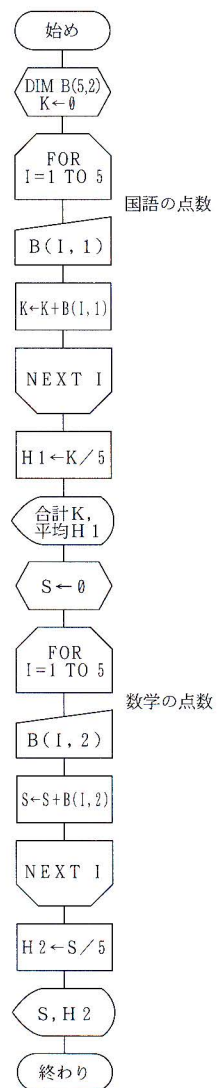
```
10  DIM  B (5, 2)
20  FOR  J=1  TO  2
30  FOR  I=1  TO  5
40  INPUT  B (I, J)
50  NEXT  I
60  S=0
70  FOR  I=1  TO  5
80  S=S+B (I, J)
90  NEXT  I
100 PRINT"ゴウケイ=";S
110 NEXT  J
120 END
```

メ モ リ 内 容	
変数	内　容
B	B(5，2)二次元配列 B(I，J)
I	I=1〜5
J	J=1〜2
S	合　計 S=0 S=S+B(I，J)

〔例　題〕㉑

下表のような５人の生徒の成績があります。科目ごとの合計と平均点を計算するプログラムを作りなさい。

番号	1	2	3	4	5
国語	58	45	69	87	75
数学	45	25	78	82	65

■フローチャート

■プログラム例

```
10  DIM B(5,2)
20  K=0
30  FOR I=1 TO 5
40  INPUT"コクコ゛ーテンスウ=";B(I,1)
50  K=K+B(I,1)
60  NEXT I
70  H1=K/5
80  PRINT"コ゛ウケイ=";K;"ヘイキン=";H1
90  S=0
100 FOR I=1 TO 5
110 INPUT"スウカ゛クーテンスウ=";B(I,2)
120 S=S+B(I,2)
130 NEXT I
140 H2=S/5
150 PRINT"コ゛ウケイ=";S;"ヘイキン=";H2
160 END
```

メ　モ　リ　内　容	
変数	内　　　容
B( )	国語の点数　数学の点数
H 1	H 1＝K／5　国語の平均
H 2	H 2＝S／5　数学の平均
K	K＝0　K＝K＋B(I，1)　国語の合計点
S	S＝0　S＝S＋B(I，2)　数学の合計点
I	I＝1〜5

■プログラムの解説

10　２次元配列、５行・２列を指定します。

20　国語の合計を求めるためにK＝0としておきます。

30〜60　Iを１から５へと順に進めて、国語の点数を順次入力します。50行で合計を求めるために累計しています。

70　５人分の国語の平均点を求めます。

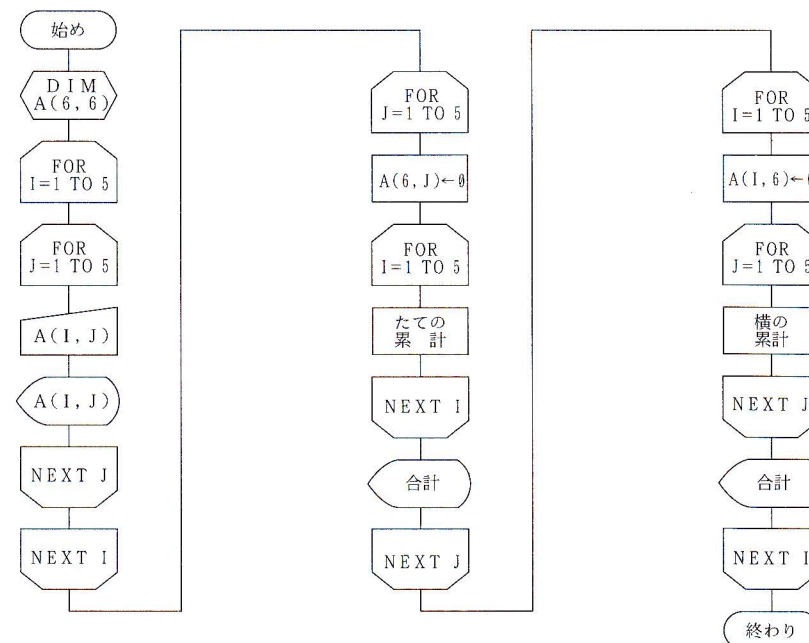
80　合計と平均値を表示します。

90〜150　今度は数学の計算を行います。国語の場合とほぼ同じです。
ただ、国語はB(I，1)の変数を使いましたが、数学ではB(I，2)の変数を使います。

〔例　題〕㉒

下のような数値の表があります。
数値を、1、2、3、4、5
11、12、13、14、15
と順次入力していき、たて、横の合計を求めるプログラムを作りなさい。

	1	2	3	4	5	横の合計
	1	2	3	4	5	
	11	12	13	14	15	
	21	22	23	24	25	
	31	32	33	34	35	
	41	42	43	44	45	
たての合計						

■フローチャート

■プログラム例

```
10  DIM A(6,6)
20  FOR I=1 TO 5
30  FOR J=1 TO 5
40  INPUT A(I,J)
50  PRINT"A(";I;",";J;")=";A(I,J)
60  NEXT J
70  NEXT I
80  FOR J=1 TO 5
90  A(6,J)=0
100 FOR I=1 TO 5
110 A(6,J)=A(6,J)+A(I,J)
120 NEXT I
130 PRINT"A(6,";J;")=";A(6,J)
140 NEXT J
150 FOR I=1 TO 5
160 A(I,6)=0
170 FOR J=1 TO 5
180 A(I,6)=A(I,6)+A(I,J)
190 NEXT J
200 PRINT"A(";I;",6)=";A(I,6)
210 NEXT I
220 END
```

メ モ リ 内 容	
変 数	内 容
A( )	A(6,6) 2次元配列
A(I,J)	入 力 値
A(6,J)	たての合計
A(I,6)	横の合計
I	I=1～5
J	J=1～5

■プログラムの解説

10 2次元配列を指定します。
20～70 データを入力し表示するループです。
90 A(6,J)=0はたての合計です。計算前なので0を代入します。
160 A(I,6)=0は横の合計です。計算前なので0を代入します。
40行、50行目について、実行例で説明をします。

```
RUN ↵      ?
1   ↵      A(1.,1.)=1.
           ?
```



```
2   ↵      A(1.,2.)=2.
           ?
3   ↵      A(1.,3.)=3.
           ?
4   ↵      A(1.,4.)=4.
           ?
5   ↵      A(1.,5.)=5.
           ?
11  ↵      A(2.,1.)=11.
           ?
12  ↵      A(2.,2.)=12.
           ?
```

このように、40行と50行目は、何行何列目にどのような数値が入ったかを確認できるようにしたものです。
110行目のたての累計計算は、下に示すような形でコンピュータは計算しています。

A(6,J)=A(6,J)+A(I,J)		J=1のとき	J=2のとき	J=3のとき	J=4のとき	J=5のとき
たての	初め	A(6,1)=0	→A(6,2)=0	→A(6,3)=0	→A(6,4)=0	→A(6,5)=0
累計計	Iは1	A(6,1)=1	A(6,2)=2	A(6,3)=3	A(6,4)=4	A(6,5)=5
算の途	Iは2	A(6,1)=12	A(6,2)=14	A(6,3)=16	A(6,4)=18	A(6,5)=20
中結果	Iは3	A(6,1)=33	A(6,2)=36	A(6,3)=39	A(6,4)=42	A(6,5)=45
	Iは4	A(6,1)=64	A(6,2)=68	A(6,3)=72	A(6,4)=76	A(6,5)=80 ⑤
最終結果(合計)	Iは5	A(6,1)=105 ①	A(6,2)=110 ②	A(6,3)=115 ③	A(6,4)=120 ④	A(6,5)=125

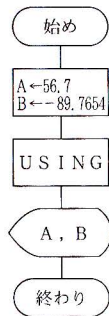
A(6,J)=A(6,J)+A(I,J)の算術式のIとJでは、繰り返しループ(FOR～NEXT文)によって上の表のような順序 ①→②→③→④→⑤ で計算が行われています。
180行目の横の累計計算も同様に、下に示すような形でコンピュータは計算しています。

A(I,6)=A(I,6)+A(I,J)		I=1のとき	I=2のとき	I=3のとき	I=4のとき	I=5のとき
横の累	初め	A(1,6)=0	→A(2,6)=0	→A(3,6)=0	→A(4,6)=0	→A(5,6)=0
計計算	Jは1	A(1,6)=1	A(2,6)=11	A(3,6)=21	A(4,6)=31	A(5,6)=41
の途中	Jは2	A(1,6)=3	A(2,6)=23	A(3,6)=43	A(4,6)=63	A(5,6)=83
結果	Jは3	A(1,6)=6	A(2,6)=36	A(3,6)=66	A(4,6)=96	A(5,6)=126
	Jは4	A(1,6)=10	A(2,6)=50	A(3,6)=90	A(4,6)=130	A(5,6)=170 ⑤
最終結果(合計)	Jは5	A(1,6)=15 ①	A(2,6)=65 ②	A(3,6)=115 ③	A(4,6)=165 ④	A(5,6)=215

演 習 問 題 7 を行ってください。

STEP 9 USING（ユージング）
PRINT USING

〔例　題〕㉓

A＝56.7、B＝−89.7654の2つの数値において、それぞれ小数点以下第3位で桁をそろえて表示するプログラムを作りなさい。

■フローチャート

■プログラム例

```
10  A＝56.7：B＝−89.7654
20  USING"####.###"
30  PRINT  A；B          ←── 20行で桁数の指定をします。
40  END
```

■実行手順

RUN　モード
RUN ↵

表示例

56.700　−89.765

〔例　題〕㉔

B＝−10、C＝50.8803の2つの数値において、文字と数値をそれぞれ表示するようにプログラムを考えなさい。小数点のあるCについては小数点以下2桁まで表示するものとします。

■フローチャート（省略）

■解　　説

USINGで数値を指定するときは＃マークを使いますが、文字を指定するときは＆マークを使います。文字の数だけ＆マークを続けて指定します。

■プログラム例

```
10  B＝−10：C＝50.8803
20  PRINT  USING"&&&###"；"B＝"；B；"␣C＝"；USING"###.#
    #"；C
30  END
```

・&&&は"B＝"と"␣C＝"で2回使用しています。

実行表示例

B＝　−10　C＝　50.88

(注)

桁数指定を解除したいときは、解除したい行にUSINGのみを指定します。USINGを指定した行以降は桁数指定が解除されます。

STEP 10 MID$（ミッド・ドル）………中間の文字を取り出す
LEN（レングス）……………………文字列の文字を数える
VAL（バリュー）……………………文字から数字へ変換する

〔例　題〕㉕

2進数を10進数に変換するプログラムを作りなさい。

■解　　説

今までは数値だけを扱ってきましたが、これから文字列に関する練習をやってみましょう。コンピュータは数字か"文字"かをはっきり指示してやらなければ動きませんが、逆に言うと人間にはできないこともやってくれます。

たとえば、2進数の10010110は10進数ではいくつになるでしょうか。

```
1    0    0    1    0    1    1    0
‖         ‖         ‖    ‖
2^7  +    2^4  +    2^2 + 1^1
128  +    16   +    4  +  2        ＝150
```

このような演算を上の桁から実行し、加算することにします。

■プログラム例

```
 10："A"：CLEAR：
     DIM  B$(0)
 20：INPUT  B$(0)
 30：L＝LEN  B$(0)
 40：N＝1
 50：C$＝MID$(B$(0),
     N, 1)
 60：E＝VAL  C$
 70：D＝E＊2∧(L−1)＋D
 80：N＝N＋1：L＝L−1
 90：IF  L＝0  THEN
     110
100：GOTO  50
110：PRINT  D
120：GOTO  10
```

10　最初にDIM B$(0)と配列宣言をして、代入する文字数を16に設定します。つまり、入力できる2進数は最大16桁までになります。B$だけにすると、最大7文字しか入力できません。(153ページ 変数の長さの項参照)
$(ドル)のついた変数が文字変数です。

30　LEN　B$(0)は文字の数を数えます。10010110なら文字の数は8です。

50　C$＝MID$(B$(0), N, 1)

60　VAL　C$では文字（C$）を数字に変換して次の計算に備えます。

80　Nはパラメーターです。1つずつ加えて行けば、B$(0)の文字列の左から1つずつ動かすことになります。L＝0で最後の桁ですから計算を止めて、110ラインで答を表示します。
11111111＝255，1111111111111111＝65535　となることを確認してください。

STEP 11 CHR$（キャラクタドル）………数をアスキー文字に変える
STR$（ストリングドル）………数を文字に変える

〔例 題〕㉖

10進数を2進数に変換するプログラムを作りなさい。

■解　説

10進数を2で割っていき、その余りを算出順に下位から順に並べて行くと2進数になります。たとえば10進数の43では、

■プログラム例

```
200："B"：CLEAR ：
     DIM B$(0)
210：INPUT N
220：I＝INT（N／2）
230：J＝N－2＊I
240：B$（0）＝CHR$（
     J＋48）＋B$（0）
250：IF I＞0 THEN
     N＝I：GOTO 220
260：PRINT B$（0）
270：GOTO 200
```

200　例題㉕と同じくDIM B$(0)と配列宣言して、答の2進数が16桁まで表示できるようにします。

220　Nを2で割ります。また、250行でNに新しいI（商）を代入し繰り返します。

230　Jは余りです。1か0です。

240　CHR$は、ASC（アスキー：次ページ参照）の逆で、10進数のアスキーコードを文字や記号（0～9の数字を含みます）に変換します。
CHR$（J＋48）のように、10進数のアスキーコードに変換するため、Jに48を加えます。
本機では、CHR$、ASC（アスキー）が使えるので活用してください。

250　I（商）が0になるまでNに代入して220行に戻ります。各桁の余りをB$(0)に入れていき、文字列を完成させます。

215行にR$＝STR$Nを追加し、260行をPRINT R$；"－>"；B$(0)としてみてください。

N（数）が文字に変わって表示されます。

このプログラムでは最大で65535＝1111111111111111　までです。

アスキー（ASCII）文字コード表

10進数	文　字
⋮	⋮
48	0
49	1
⋮	⋮
54	6
⋮	⋮
65	A
66	B
67	C
68	D
69	E
70	F
⋮	⋮

STEP 12 ASC（アスキー）………文字を数値で表す
論理演算子………………………ホント（真）かウソ（偽）か？

〔例 題〕㉗

10進数を16進数に、また16進数を10進数に変換するプログラムを作りなさい。

■解　説

本機は&Hh（hは16進数で0～FFFFFFFFまで）で16進数を10進数に、HEX$で10進数を16進数に変換できます。ここでは、2つともBASICで作ってみます。

①10進数→16進数変換

```
300："C"：INPUT N
310：H$＝""
320：I＝ INT （N／16）
330：J＝N－16＊I
340：H$＝CHR$ （J＋48
     －（J＞9）＊7）＋H$
350：IF I＞0 THEN
     N＝I：GOTO 320
360：PRINT H$
370：GOTO 300
```

10進→16進変換は、10進→2進変換の方法とほとんど同じです。例題㉖と同じように、10進数を16で割り、商と余りを求めることを繰り返します。

たとえば、$(246)_D＝(F6)_H$では

（Dは10進を、Hは16進を示します。）

商	余り	16進数	アスキーコード
$\frac{246}{16}=15$	J＝6	6	54
$\frac{15}{16}=0$	J＝15	F	70

答はH$＝F6です。

340行で（J＞9）はJ＞9がホントなら－1、ウソなら0という答になります。したがって、J＝6ならCHR$（6＋48－0×7）＝CHR$54、J＝15ならCHR$（15＋48－（－1）×7）＝CHR$70です。CHR$↔ASCの意味をよく考えてみてください。

②16進数→10進数変換

```
400："D"：INPUT H$
410：N＝0
420：FOR I＝1 TO
     LEN H$
430：P$＝MID$ （H$,
     I, 1）
440：J＝ASC P$－48
450：J＝J＋（J＞9）＊7
460：N＝N＊16＋J
470：NEXT I
480：PRINT N
490：GOTO 400
```

16進→10進変換は、2進→10進変換の方法と同じで上の桁から始めればよいでしょう。

たとえば、$(AD3)_H＝(2771)_D$では

（A　D　3）$_H$＝$(2771)_D$

10×16＋13＝173

173×16＋3＝2771

プログラムはこの計算を行っています。

440行でJ＝ASC"A"－48＝65－48＝17

450行でJ＝17＋－1×7＝10となります。

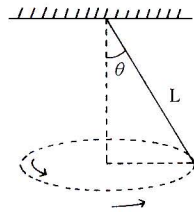
以下をよくたどってみてください。

例題㉕、㉖、㉗をそのまま実行すれば、２進数 ⇄ 10進数 ⇄ 16進数　で基本的な変換
（GOTO "A"↵　GOTO "C"↵ ／ GOTO "B"↵　GOTO "D"↵）

はできますが、ここで少し工夫してもらいたいことがあります。

1．演算可能な桁数以上の数を入力した、などのミスを防ぐにはどうすればよいでしょうか。
2．２進数⇄10進数変換で、ここでは16ビットまでの１の補数で表していますが、２の補数で表すにはどうすればよいでしょうか。
3．表示を見やすいように工夫してください。答の表示では、10進数をＳＴＲ＄を使って"文字"に直したほうがよいでしょう。特に250、350でＮの数値が変化していることに注意してください。

演習問題　８　９　を行ってください。

〔例　題〕㉘　71ページの例題①をプログラム化した例題

①左図の円すい振り子の周期をＴ、糸の長さをＬ、糸の鉛直となす角をθとすると、

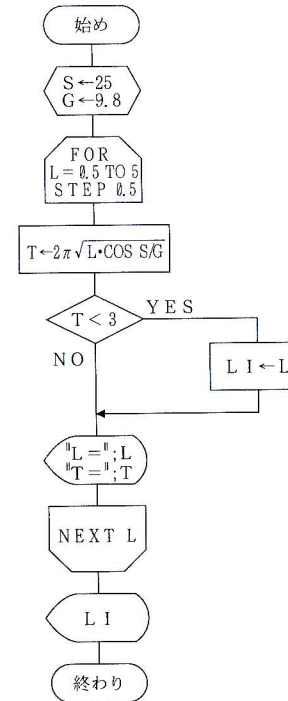
$$T = 2\pi\sqrt{\frac{L\cdot\cos\theta}{g}}$$

の関係があります。今、糸の長さＬを0.5〔m〕から５〔m〕まで、0.5〔m〕刻みで変化させたときのＬとＴの変化を表示するプログラムを作りなさい。ただし、g＝9.8〔m/sec²〕、θ＝25°とします。

②また、それぞれを表示した後で、Ｔの値が３秒を超えないＬの最も大きい値を表示するプログラムを作りなさい。

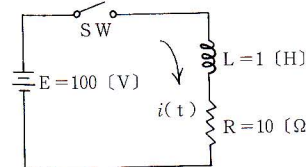
■プログラム例

```
10  REM フリコ
20  S=25:G=9.8
30  FOR L=0.5 TO 5 STEP 0.5
40  T=2*PI*SQR(L*COS S/G)
50  IF T<3 THEN LI=L
60  PRINT"L=";L;USING"##.##";" T=";T
65  FOR J=1 TO 1000:NEXT J
70  USING
80  NEXT L
90  PRINT"T<3---L=";LI;"(M)"
100 END
```


■フローチャート

■結　果

```
L=0.5   T= 1.35
L=1.    T= 1.91
L=1.5   T= 2.34
L=2.    T= 2.70
L=2.5   T= 3.02
L=3.    T= 3.30
L=3.5   T= 3.57
L=4.    T= 3.82
L=4.5   T= 4.05
L=5.    T= 4.27

T<3---L=2. (M)
```

〔例　題〕㉙　72ページの例題③をプログラム化した例題

（a）左図の回路において、スイッチＳＷを入れてから0.1秒ごとに１秒までの間に流れる電流を表示するプログラムを作りなさい。ただし、時間と電流を同一行に表示するようにしなさい。

$$i(t) = \frac{E}{R}\left(1 - e^{-\frac{R}{L}t}\right)$$

（b）次に、抵抗Ｒを10〔Ω〕から10〔Ω〕ずつ50〔Ω〕まで変化させたとき、回路を流れる電流が、$i = \frac{E}{R}$ の62％から65％の範囲になる時間を求めるプログラムを作りなさい。ただし、時間は０秒から0.2秒まで0.002秒間隔で調べなさい。

■プログラム（a）

```
10 REM R-L
20 INPUT"R=";R,"L=";L,"E=";E
30 FOR T=0 TO 1 STEP 0.1
40 I=E/R*(1-EXP(-R*T/L))
50 PRINT"T=";T;"␣I=";USING"##.
   ###";I
55 FOR J=1 TO 1000:NEXT J
60 USING
70 NEXT T
80 END
```

■結　果（a）

```
T=0.   I= 0.000
T=0.1  I= 6.321
T=0.2  I= 8.646
T=0.3  I= 9.502
T=0.4  I= 9.816
T=0.5  I= 9.932
T=0.6  I= 9.975
T=0.7  I= 9.990
T=0.8  I= 9.996
T=0.9  I= 9.998
T=1.   I= 9.999
```

■プログラム（b）

```
10 REM R-L
20 INPUT"L=";L,"E=";E
30 FOR R=10 TO 50 STEP 10
40 FOR T=0 TO 0.2 STEP 0.002
50 I=E/R*(1-EXP(-R*T/L))
60 IF (I>=E*0.62/R) AND (I<=E*
   0.65/R) THEN PRINT"TR(";R;")
   =";T;"(S)":GOTO 80
70 NEXT T
80 NEXT R
90 END
```

■結　果（b）

```
TR(10.)=0.098 (S)
TR(20.)=0.05  (S)
TR(30.)=0.034 (S)
TR(40.)=0.026 (S)
TR(50.)=0.02  (S)
```

3．構造化BASIC命令の使いかた

構造化BASIC命令と呼ばれる次の4つの命令について説明します。これらの命令はプログラムをより見やすくわかりやすくするために考案されたものです。

(1)　判断（IF～THEN～ELSE～ENDIF）

IF～THEN～ELSEは"もし～ならば～"や"もし～でなければ～"という意味で、いろいろな判断を行う命令です。

STEP④で説明した"IF～THEN～ELSE"（1行形式）の他に"IF～THEN～ELSE～ENDIF"（ブロック形式）を用いて判断することができます。基本的な動作は1行形式のIF文と同じです。

ブロック形式のIF文は次の形で用います。


```
IF 条件式 THEN
 実行文
[ELSE
 実行文]
ENDIF
```

次の1行形式の命令文

```
10 IF A>=10 THEN PRINT A ELSE A=A+1
```

は、ブロック形式では次のように書き表されます。

```
10 IF A>=10 THEN
20     PRINT A
30 ELSE
40     A=A+1
50 ENDIF
```

IF文が成立すればIF文とELSEの間の命令を実行し、終了後ENDIFの次の行を実行します。成立しなければELSEが書かれている行の次の行から実行します。実行文が複雑になる場合は、ブロック形式にするとわかりやすくなります。

(注)　ブロック形式か1行形式かの判断は、IFが行頭にあり、なおかつTHENの後に実行文が書かれているかどうかで行います。

(2)　繰り返し（REPEAT～UNTIL、WHILE～WEND）

STEP⑤で説明した"FOR～NEXT"は繰り返し回数を最初に指定するのに対し、次の2つの命令はいつ終わるかわからない繰り返しに使用します。

"REPEAT～UNTIL"は条件が成立するまで繰り返し実行（条件を満足すると終了）するのに対し"WHILE～WEND"は条件が成立している間繰り返し実行します。

REPEAT～UNTIL

一度ループ内を実行し、最後に条件判断して繰り返します。

REPEAT～UNTIL文は次の形で用います。

```
REPEAT
  ～
UNTIL 条件式
```

WHILE～WEND

ループに入る前に条件判断するため、一度もループ内を実行せずに終わる場合もあります。

WHILE～WEND文は次の形で用います。

```
WHILE 条件式
  ～
WEND
```

(3) 振り分け(SWITCH～CASE～DEFAULT～END SWITCH)

ON～GOTO命令は数値だけを扱うのに対し、SWITCH～CASE命令では、文字も扱うことができます。したがって、入力された変数をそのまま比較し振り分けるには、SWITCH～CASEの命令が便利です。たとえば、電子電話帳などのプログラム作成に利用すると便利です。

SWITCH～CASE～DEFAULT～ENDSWITCH文は次の形で用います。

```
SWITCH 変数
  CASE {式 / 文字列}
    実行文
 [CASE {式 / 文字列}
    実行文]
 [DEFAULT
    実行文]
ENDSWITCH
```

4．スクリーンエディタについて

BASICのPROモードやTEXTのエディット中ではスクリーンエディタ機能が働き、プログラムの修正や編集が容易になりました。ただし、BASICのRUNモードなどでは働きません。106ページで説明した円の面積を求めるプログラムを、スクリーンエディタ機能を使って球の体積を求めるプログラムに変更してみます。

LIST命令でプログラムリストを表示させます。

キー操作	表示部
PROモードにします。 LIST[↵]	10:INPUT R 20:S= PI *R∧2 30:PRINT S 40:END
[▶][▼]V[▶]4[/]3[*][π] [*]R[y^x]3	10 INPUT R 20 V=4/3*PI*R∧3_ 30 PRINT S 40 END
[▼][◀]R[,]V	10 INPUT R 20 V=4/3* PI *R∧3 30 PRINT R, V_ 40 END

キー操作	表示部
[↵] 修正を終了します	10 INPUT R 20 V=4/3* PI *R∧3 30 PRINT R, V 40 END

スクリーンエディタ機能では1行ごとの修正だけではなく、プログラム全体を平面的に考えて修正することもできます。1行ごとに[↵]を押す必要はありません。ただし、修正の最後は[↵]を押してください。[↵]を押さずに[CLS]を押すと、最後の行が修正されません。

なお、異なるプログラム行にカーソルを移動させたときや[BASIC]を押したときも、プログラム修正は行われます。

次にスクリーンエディタの中での主なキーの動作について説明します。

[▶]……1桁右にカーソルを移動
　　行末にあるときは、次の行の先頭に移動

[◀]……1桁左にカーソルを移動
　　先頭にあるときは、前の行の行末に移動

[▼]……桁位置を変えずに1行下にカーソルを移動
　　行末よりも右側になるときは行末に移動

[▲]……桁位置を変えずに1行上にカーソルを移動
　　行末よりも右側になるときは行末に移動

これらのキーは押し続けると連続して送っていきます。

[↵]……プログラムの修正を終了します。

①プログラム行の削除

プログラムの任意の行を削除するときは、[CLS]を押して表示を消した後、不要なラインナンバーの番号を入力して[↵]を押すか、またはラインナンバーだけを残して命令をすべて削除します。

〈例〉　不要なラインナンバー[↵]

なお、ラインナンバーも含めて削除（[SHIFT]＋[DEL]の操作）すると、削除したことにはなりません。連続した複数行のプログラムを削除するときは、DELETE命令を使用することもできます。

キー操作	表示部
LIST[↵]	10:INPUT R 20:V=4/3* PI *R∧3 30:PRINT R, V 40:END
[◀][◀]	10 INPUT R 20 V=4/3* PI *R∧3 30 PRINT R, V 40 END

キー操作	表示部
SHIFT + DEL （10回押す）	_ 20 V=4／3＊ PI ＊R∧3 30 PRINT R, V 40 END
▼ 元の10行目の内容があらわれます。 （削除したことにはなりません。）	10 INPUT R 20 V=4／3＊ PI ＊R∧3 30 PRINT R, V 40 END

②プログラム行の複写（追加）

プログラムリスト表示中に、ラインナンバーを書き換えるとプログラム行の複写ができます。元のプログラム行は残っており、複写したプログラム行の内容は変更することもできます。
複写（ラインナンバーを書き換える）した後、異なるプログラム行にカーソルを移動させる（↵または、▼、▲の操作）と、プログラムは番号順に並び替わって表示されます。
新しいプログラム行を追加するときは、上のように複写してプログラム行を追加するか、または、CLSを押して表示を消した後、新しいプログラム行を追加します。なお、プログラムリストの最終行の次の行からはプログラム行の追加ができます。

連続した複数行のプログラムを複写するときは、LCOPY命令を使用することもできます。

BASICによるプログラム演習問題

演習問題を始める前に

これからBASIC言語の演習問題を手がけていきましょう。演習問題を始める前に、次のことをよくまもって解答してください。以後、演習問題は下の1、2、3項にもとづいて行うこととします。

1．問題文をよく読んでプログラムの手順となる「流れ図」（フローチャート）を作る。
2．プログラムリストを別紙に書く（ただしプリンタをお持ちの方は除く）。
3．メモリ内容表を作成する。

（プログラムリスト参考例）　　タイトルを必ず書く。

（メモリ内容表参考例）

メモリ内容表	
変数	内容
A	三角形の一辺
B	三角形の一辺
T	$T=\frac{a+b+c}{2}$

●解答は先生にお聞きください。（指導マニュアルに解答例を記載しています。）

演習問題 1

1．半径Rを入力して円周ℓを求めるプログラムを作りなさい。
半径Rは2〔cm〕、4〔cm〕、6〔cm〕、8〔cm〕、10〔cm〕を入力するものとします。

R〔cm〕	2	4	6	8	10
ℓ〔cm〕					

2．三角形の底辺Lと高さHを入力して面積Aを求めるプログラムを作りなさい。
L，Hの値は、それぞれ表の数値とします。

L〔cm〕	5	10	15	20	25
H〔cm〕	3	6	9	12	15
A〔cm^2〕					

3．水X〔g〕の中へ塩Y〔g〕を入れたときの濃度N〔%〕を求めるプログラムを作りなさい。

X	1000	1000	1000
Y	5	10	15
$N=\frac{100\times Y}{X+Y}$			

4．数X，Y，Zを入力してXY＋YZ＋ZXを求めるプログラムを作りなさい。

X	1	－２	3	－４	5
Y	6	－７	8	－９	10
Z	－11	12	－13	14	15
結果					

5．数X，Y，Zを入力して $(XY^2+YZ^2+ZX^2)$ を求めるプログラムを作りなさい。
X，Y，Zの値は演習問題１の４．の値を用いなさい。

6．数X，Yを入力して $(X/6-Y/3)^2$ を求めるプログラムを作りなさい。

X	83	84	85	86	87
Y	19	18	17	16	15
結果					

演習問題　2

1．123.619を入力して、(１) 切り捨て (２) 四捨五入をして、小数第２位まで求めるプログラムを作りなさい。

2．数X，Yを入力してX＋Yを計算し、プラスなら１、マイナスなら－１、０なら０と表示するプログラムを作りなさい。

数値例

X	1	3	5	4	0
Y	2	－５	6	－４	－３
結果					

3．数X，Y，Zを入力してX＊Y＊Zを計算し、プラスなら１、マイナスなら－１、０なら０と表示するプログラムを作りなさい。

数値例

X	1	－１	－１	0	－５
Y	2	－２	2	2	2
Z	3	－３	－３	5	－３
結果					

4．２桁の乱数を２個作り、それらを表示し、その乱数の差の絶対値を求めるプログラムを作りなさい。

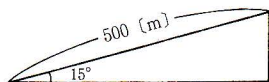
5．２桁の乱数を２個作り、それらを表示し、その乱数の和、差、積、商を求めるプログラムを作りなさい。

6．X度を入力して $\sin(X)+\sqrt{3}\cdot\cos(X)$ の値を求めるプログラムを作りなさい。

数値例

X	15°	30°	45°
結果			

7．sin(A＋B)＝sin(A)cos(B)＋cos(A)sin(B) です。右辺と左辺を別々に計算して表示するプログラムを作りなさい。　A＝15°、B＝30° とします。

8．下図のように15度の坂道を歩くと何m高くなっているかを求めるプログラムを作りなさい。

演習問題　3

1．数A，B，Cを入力してA＞BかつB＞CならA＊B＊Cの計算を、A＞BかつB≦CならA＋B＋Cの値を、A≦BならA／B＋Cの値を求めるプログラムを作りなさい。

2．最初の日に１円、次の日に２円、次の日に４円と順に毎日倍額ずつ貯金をすると100万円を超える日は何日目であるかを求めるプログラムを作りなさい。

3．１＋２＋３＋………＋Xの合計が初めて200を超えるXの値を表示するプログラムを作りなさい。

4．１，３，５，７，９，…………，Xまでの和を求め、和が1000を超えないXの最大値を表示するプログラムを作りなさい。

5．X，Yを入力し、X＝１でY＝２なら〃A〃と、X＝３でY＝４なら〃B〃と、X＝５でY＝６なら〃C〃と表示し、上記以外の数字の組み合せを入れたときはプログラムの最初に戻るプログラムを作りなさい。

6．２つの数XとYを入力し、両方マイナスのときは $\sqrt{X*Y}$ を、どちらか一方がプラスのときはX＊Yを、両方プラスのときはX／Yのそれぞれの値を表示するプログラムを作りなさい。

7．数値を10個入力して合計と平均を求めるプログラムを作りなさい。

演習問題　4

1．自然数１から100までの和を求めるプログラムを作りなさい。
S＝１＋２＋３＋………＋100

2．自然数MからNまでの和と平均を求めるプログラムを作りなさい。M、Nの値は入力するものとしM＜Nとします。
S＝M＋……………＋N

3．$Y = 3X^3+2X^2+X+15$ において、Xの値を－１０から１０まで、０．５刻みで変化させてYの値を求めるプログラムを作り、結果を表にしなさい。

4．ⓐ　X（単位は度）を０度から３６０度まで１０度刻みで変化させて sinXを求めるプログラムを作り０度から３６０度までの表を完成しなさい。
ⓑ　cosXについてもⓐと同じことをしなさい。
ⓒ　tanXについてもⓐと同じことをしなさい。ただし、tan 90°と tan 270°のときは無限大となりエラーになりますので、〃＊＊＊＊〃のように表示させなさい。

5．数Xを入力してXが正なら sinXを、Xが０なら cosXを、Xが負なら tanXを求めるプログラムを作りなさい。

6．AAB＋BB＝BAAつまり、(100×A＋10×A＋B)＋(10×B＋B)＝(100×B＋10×A＋A) となるAとBを求めなさい。A、Bは１桁の整数とします。

7．ABA×B＝BCBつまり、(100×A＋10×B＋A)×B＝(100×B＋10×C＋B) となるA、B、Cを求めるプログラムを作りなさい。

8．三角形の辺A、B、Cにおいて、それぞれを１から２０まで変化させたとき、直角三角形となる組み合せをすべて求めるプログラムを作りなさい。

9．$Y = 6X^2-5X-9$ の式において、$-10 \leqq X \leqq 10$ の範囲でYの最大値を求めるプログラムを作りなさい。Xは０．２刻みとします。

10. データを10個入力して最大値と最小値を求めるプログラムを作りなさい。
　　５、７、－６、８、15、－４、－８、９、13、12

演習問題　5

1．データ２、４、６、８、10を読んで、その積を求めるプログラムを作りなさい。
2．次のデータを、200から順に引いて、答えを表示するプログラムを作りなさい。
　　データ　２４、８、２９、３５、１０
3．次のデータのうち、最初の数をそれ以降のデータで割算をして、答えを表示するプログラムを作りなさい。
　　データ　１９８２、６、１２、２５
4．Ａ＝57、Ｂ＝８、Ｃ＝14となるように、データ57、８、14を読み込んでそれを表示し、Ａ＊Ｃ／Ｂの答えを表示するプログラムを作りなさい。
5．データ５、６、７、８、９を読んで、それらをそれぞれＡ、Ｂ、Ｃ、Ｄ、Ｅに割りあてて、Ａ＊Ｂ、（Ａ＊Ｂ）／（Ｃ＋Ｄ）、（Ａ＊Ｅ）－（Ｂ＋Ｃ＋Ｄ）の答えを表示するプログラムを作りなさい。
6．10人の身長を測定した結果、次のようになりました。10人の身長の測定データを読み込んで、平均身長を求めるプログラムを作りなさい。
　　データ　165.5，170.5，172.3，168.4，182.6，
　　　　　　159.9，174.8，167.6，177.8，173.6
7．次のデータを読んでＡ＝35、Ｂ＝29、Ｃ＝８、Ｄ＝８、Ｅ＝49、Ｆ＝19、Ｇ＝８、Ｈ＝49となるようにそれぞれ表示するプログラムを作りなさい。
　　データ　３５、２９、８、４９、１９
8．10人に数学のテストを行ったところ、次のような結果でした。それぞれの点数を読んでから10人の平均点を求め、さらに平均点に最も近い点数を見つけ出し、平均点と平均点に最も近い点数を表示するプログラムを作りなさい。

番　　号	1	2	3	4	5	6	7	8	9	10
点　数(点)	20	40	50	35	70	80	68	55	90	83

演習問題　6

1．数Ｘを入力してＸ＝１のときX^2、Ｘ＝２のとき$\sqrt{X}$、Ｘ＝３のときX^3、Ｘ＝４のときX^4を表示するプログラムを作りなさい。
2．数Ｘを順次入力して、０が入力されたらそれまでの数の和と平方和を求めるプログラムを作りなさい。
3．数Ｘを入力して$(3X^2-3)+(3X^2-3)^2-6X$を求めるプログラムを作りなさい。
4．数Ｘ，Ｙ，Ｚを入力して$(6X^2+1)-(6Y^2+1)+(6Z^2+1)$を求めるプログラムを作りなさい。

演習問題　7

1．〔例題〕⑳において、Ｍ個、Ｎ列のようなデータならプログラムをどのように作り替えたらよいか。そのプログラムを作りなさい。
2．〔例題〕㉑において、Ｎ人の生徒でＫ科目の場合の合計と平均を求めるプログラムに作り替えなさい。
3．〔例題〕㉒において、データ表の数値を入力した後、斜めのライン、1、12、23、34、45と５、14、23、32、41の合計を求めるプログラムを作りなさい。
4．下の表のような10人の生徒の得点を入力し、合計および平均、母標準偏差σと偏差値Ｔｉを求めるプログラムを作りなさい。

番　号	1	2	3	4	5	6	7	8	9	10
得　点	85	90	78	85	95	68	59	74	100	66

各人の得点を$X_1, X_2 \cdots X_n$（ｎは人数）とすると、

合計　$\Sigma Xi = X_1+X_2+\cdots+X_n$　　平均$\overline{X} = \dfrac{\Sigma Xi}{n}$（ここではＨと表現します）

母標準偏差　$\sigma = \sqrt{\dfrac{\Sigma Xi^2 - nH^2}{n}} = \sqrt{\Sigma Xi^2/n - H^2}$

偏差値　$Ti = 10 \times \dfrac{Xi - H}{\sigma} + 50$

で求められます。

5．４の問題のデータを用いて低い点数から順に並べ替えをするプログラムを作りなさい。

演習問題　8

1．文字列"ＡＢＣＤＥＦＧＨＩＪ"を入力してＡ、Ｃ、Ｅ、Ｇ、Ｉと１つ飛びにばらばらに表示するプログラムを作りなさい。
2．文字列"ＡＢＣＤＥＦＧＨＩＪ"を入力してＡ、ＡＢ、ＡＢＣ、ＡＢＣＤ、ＡＢＣＤＥと前の表示よりも１文字ずつ増えて表示するプログラムを作りなさい。
3．文字列"ＡＢＣＤＥ５４３２１"を入力して"１２３４５ＥＤＣＢＡ"と表示するプログラムを作りなさい。
4．文字列　Ｉ＄＝"Ｉ"、Ａ＄＝"␣ＡＭ"、Ａ１＄＝"␣Ａ"、Ｃ＄(0)＝"␣ＣＯＭＰＵＴＥＲ"を代入して、Ｉ　ＡＭ　Ａ　ＣＯＭＰＵＴＥＲと表示するプログラムを作りなさい。
5．文字列　Ｘ＄＝"Ｉ　ＡＭ　Ａ　ＣＯＭＰＵＴＥＲ"を入力して、その中の"ＣＯＭ"だけ表示するプログラムを作りなさい。

演習問題　9

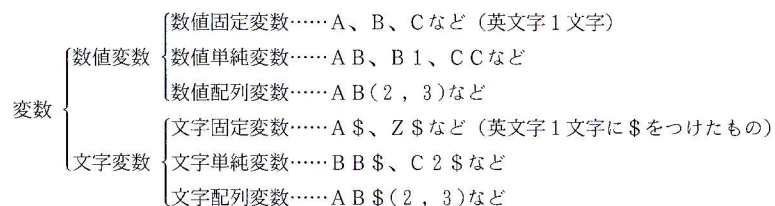
1．２つの文字列Ａ＄＝"ＭＡＸＡＢＣ"、Ｂ＄＝"ＸＹＭＩＮ"を入力して文字数の大小を比較した後、大きい方の文字の頭文字３文字を表示するプログラムを作りなさい。
2．各自、自分自身の姓と名をそれぞれローマ字読みで入れ、その長さが何文字かを調べ、次に名を先に、姓を後に表示するプログラムを作りなさい。

5．変数の種類と使いかた

これまで変数を使ってプログラムを作ってきましたが、ここで変数の種類と使いかたについて説明します。

(1)　変数の種類

変数には大きく分けて、数値変数と文字変数の2種類があります。それらは、さらに固定変数、単純変数、配列変数に分類されます。

変数
- 数値変数
  - 数値固定変数……A、B、Cなど（英文字1文字）
  - 数値単純変数……AB、B1、CCなど
  - 数値配列変数……AB(2，3)など
- 文字変数
  - 文字固定変数……A$、Z$など（英文字1文字に$をつけたもの）
  - 文字単純変数……BB$、C2$など
  - 文字配列変数……AB$(2，3)など

【変数の名前】

変数の名前には、A～Zのアルファベット1文字か、あるいはAA、BC、A5などのように文字や数字を組み合わせた2文字が使用できます。

- 変数名には次のような文字は使用できません。
  ①カタカナおよび記号（ただし、$記号は文字変数を表す記号として使用されます。）
  ②予約語（予約語とは、本機がBASIC命令や関数命令などの名前として使用しているものです。）
  〈例〉　PI、IF、TO、ON、SINなど
  ③小文字のアルファベット（小文字のアルファベットは大文字に変換されます。）
- 変数は必ずアルファベットで始まっていなければなりません。たとえばA5は変数名になりますが、5Aは変数名になりません。
- 変数名に3文字以上使用してもエラーにはなりませんが、本機が変数名と判断するのは最初の2文字だけです。たとえば、変数名にKOTAE1とKOTAE2を使用すると、これらを区別できずにKOという同じ変数と判断します。
- 固定変数（1文字で指定する変数）では、数値変数と文字変数に同じ名前を使用できません。たとえば、数値変数としてAを使用しているとき、文字変数A$を使用することはできません。（この場合、後からA$に文字を代入すると、前の数値変数Aが消されます。）

【変数の長さ】

変数はその種類によって、長さ（その変数に格納できるデータの最大の長さ）が決められています。次に各変数の長さを示します。

変数の種類	変　数　の　長　さ
数　値　変　数	有効数字10桁まで（仮数部10桁、指数部2桁）
文字固定変数	7文字まで
文字単純変数	16文字まで
文字配列変数	16文字（ただし、1～255文字まで設定可能）

- それぞれの文字変数の長さを超える文字を記憶させようとした場合、超えた分の文字は無視（切り捨て）されます。

(2)　固定変数

これまでの説明やプログラム例で、変数名をA、B、C…またはA$、B$、C$…と指定して使用しましたが、これらの変数を固定変数（1文字変数）と呼びます。

固定変数は、変数として使用されるエリア（区域）がメモリ上に独立して確保されており、このエリアにはプログラムなどが書き込まれることはありません。

このエリアをデータ専用エリアと呼びます。

また、同じ名前の数値固定変数と文字固定変数（たとえばAとA$）は同じ場所が使用されます。

(3)　単純変数

単純変数は変数名をAAやB1のように2文字（あるいはそれ以上）で指定する変数です。

この変数は固定変数のように使用される場所は決まっておらず、初めて使用したときにメモリ内（プログラム・データエリア内）に自動的に確保されます。

また、同じ変数名でも数値単純変数と文字単純変数は別々に確保されるので、たとえばABとAB$を同時に使用することもできます。

(4)　配列変数（一次元配列、二次元配列）

数学などで同じ性質の複数個のデータを表すとき、たとえば、

$X_1, X_2, X_3 \cdots\cdots X_n$

のように、1つの変数名に添字をつけて表す場合があります。

BASICでも同様に、1つの変数名に（　）で囲った添字をつけて、次のように表すことができます。

X(1)、X(2)、X(3)……X(10)

このように添字をつけて表す変数を配列変数と呼びます。

配列変数を使うとデータの集計などの作業がしやすくなります。

配列変数を使用するときは、事前にDIM（ディメンジョン）命令によって、配列名とその大きさを定義（宣言）しておかなければなりません。（くわしくは、BASICの各命令の説明のDIM命令参照）

DIM命令で定義することによって、その大きさの配列変数がメモリ（プログラム・データエリア）上に確保されます。

【一次元配列】

添字が1個だけのものを一次元配列といいます。たとえば、

DIM　X(5)

を実行すると、次のように配列名Xの6個の変数（配列要素）が確保されます。

X(0)	X(1)	X(2)	X(3)	X(4)	X(5)

（注）この図は配列が横に並んでいるように書いていますが、縦に並んでいると考えても同じです。実際の使いかたは129ページを参照してください。

【二次元配列】

本機は添字を２個まで使用できます。添字が２個のものを二次元配列といいます。
一次元配列が、いうなれば横一列（または縦一列）の配列だったのに対し、二次元配列は縦と横の配列といえます。下の図は、

　　ＤＩＭ　Ｘ（３，５）

と定義したときに確保される変数（配列要素）を表したものです。24個の変数が確保されます。

X(0,0)	X(0,1)	X(0,2)	X(0,3)	X(0,4)	X(0,5)
X(1,0)	X(1,1)	X(1,2)	X(1,3)	X(1,4)	X(1,5)
X(2,0)	X(2,1)	X(2,2)	X(2,3)	X(2,4)	X(2,5)
X(3,0)	X(3,1)	X(3,2)	X(3,3)	X(3,4)	X(3,5)

【文字配列変数】

配列変数にも文字変数があり、配列名に＄（ドル）記号をつけて表します。

　　ＤＩＭ　Ｚ＄（９）　　　　　ＤＩＭ　Ｘ１＄（２，１）
　　ＤＩＭ　Ｙ＄（５，４）　　　Ｘ１＄（0，１）＝"ヨコハマ"

● 文字配列変数の拡張

文字固定変数は１個の変数に最大７文字、文字単純変数は１個の変数に最大16文字記憶できる固定長の変数でしたが、文字配列変数は１～255文字の範囲で、任意に変数の長さを指定できます。

　　ＤＩＭ　Ｃ＄（９）＊３０
　　　　　　Ｃ＄（0）～Ｃ＄（９）の各変数には、それぞれ最大30文字まで記憶できます。
　　ＤＩＭ　Ｎ＄（５，４）＊６
　　　　　　Ｎ＄（0，0）～Ｎ＄（５，４）の各変数には、それぞれ最大６文字まで記憶できます。

このように、ＤＩＭ文に＊式をつけて、変数の長さを指定します。
なお、長さを指定しない（＊式がない）ときは自動的に16文字が指定されます。

【メモリの構成と変数】

プログラム、単純変数、配列変数が記憶される領域をプログラム・データエリアと呼びます。
プログラムはプログラム・データエリアの前方から、データは後方からエリアを確保します。そのためにプログラムの長さによって変数として使える大きさが変わってきますので注意が必要です。なお、固定変数はデータ専用エリアとして別に確保されています。

次に、各変数を確保する場合に使用するバイト数およびプログラムの各命令などの占めるバイト数を記しておきますので参考にしてください。

変数の場合

変　数	変数の名前	デ　ー　タ
数値単純変数 数値配列変数	７バイト	８バイト
文字単純変数	７バイト	16バイト
文字配列変数	７バイト	指定されたバイト数

● たとえば、ＤＩＭ　Ｂ＄（２，３）＊１0と指定した場合は、次のバイト数を使用します。
　７バイト（変数名）＋10バイト（文字数）×｛（２＋１）×（３＋１）｝個＝127バイトになります。

命令の場合

構成要素	ラインナンバー	命令文、関数	↵、その他
使用バイト数	３バイト	２バイト	１バイト

● たとえば、下のプログラム行を入力した場合、次のバイト数を使用します。

　　１0　ＰＲＩＮＴ　Ａ；"Ｘ␣"↵

　３バイト（行番号）＋２バイト（命令文）＋１バイト（↵）＋６バイト（その他１×６）＝12バイト

● プログラム・データエリアの残りのバイト数（フリーエリア）はＦＲＥ命令で求めることができます。
　（くわしくは、ＢＡＳＩＣの各命令の説明のＦＲＥ命令を参照してください。）

【変数をクリア（消去）するには】

① 個々の変数の内容を消去するには、次のようにします。

〈例〉　Ａ＝0　　　｝数値変数は0を代入することにより消去します。
　　　　Ｂ（１）＝0
　　　　Ａ＄＝""　　｝文字変数は""を代入することにより消去します。
　　　　Ｂ１＄＝""　　""を代入するということは文字変数を文字が入っていない状態にするということです。この状態を Null（ヌル）と呼びます。

● 変数の内容を消去してもフリーエリアの大きさは変わりません。次のＣＬＥＡＲ命令やＥＲＡＳＥ命令を実行するとフリーエリアはふえます。

② ＣＬＥＡＲ命令

すべての変数を一度に消去するには、ＣＬＥＡＲ命令を実行します。
ＣＬＥＡＲ命令の一般形は次のようになります。

　　ＣＬＥＡＲ

この命令を実行すると、メモリ内に確保されていた配列変数や単純変数は消去され、固定変数の内容も消去されます。

● この命令ではプログラムは消去されません。配列変数や単純変数を消去することにより、フリーエリアがふえます。

③ ＥＲＡＳＥ命令

配列変数のみを消去します。
ＥＲＡＳＥ命令の一般形は次のようになります。

　　ＥＲＡＳＥ　配列名［，配列名，……］

この命令を実行すると、指定された配列名の配列変数を消去します。なお、配列名の指定では、配列の大きさ（ＢＡＳＩＣの各命令の説明のＤＩＭ命令参照）を指定する必要はありません。

６．デバッグ

プログラムを実行したときに、何かの誤りで迷走したり思わぬ結果が出たりすると、プログラムリストを調べて誤りを探しますが、それだけではなかなか見つけにくいことがあります。

このようなときは、プログラムを1行ずつ実行させ、その経過をたどりながら誤りを探していくと見つけやすくなります。
ここでは、1行ずつ経過をたどりながらプログラムを実行する方法について説明します。(この方法をトレースといいます。また、プログラムの誤りを探し、修正することをデバッグ(虫取り)といいます。)

(1) デバッグのしかた

①RUNモードにしてください。トレースによるデバッグはRUNモードで行います。
②TRON [↵] と押して、トレースモードにします。
③RUN [↵] と押してプログラムの実行を開始します。(最初の行の実行が終われば、実行した行番号を表示して実行が停止します。)
④その後は [▼] を押します。(1行だけ実行して停止します。) INPUT命令でのデータ入力は、通常のプログラム実行と同じように [↵] を押します。
⑤プログラムの実行順序の確認や、各行(各ライン)実行後の変数の内容確認などを行いながらトレースを進め、プログラムが正しく実行されているかどうかチェックします。
正しく実行されないときは、その原因を探して修正します。
⑥デバッグが終了すれば、TROFF [↵] と押してトレースモードを解除します。

次に簡単な例を示します。

〈例〉
```
10 INPUT "A=";A,"B=";B
20 C=A*2
30 D=B*3
40 PRINT "C=";C;"␣D=";D
50 END
```

[キー操作] RUNモード

キー操作	表示	
TRON [↵]		
RUN [↵]	A=_	INPUT命令実行
8 [↵] (データ入力)	B=_	
9 [↵] (データ入力)	10:	← 10行終了
[▼]	20:	← 20行終了
[▼]	30:	← 30行終了
[▼]	C=16.　D=27.	← PRINT命令実行
	40:	← 40行終了
[▼]	>	実行終了　(プロンプト表示)

トレース中で行番号を表示しているときは、マニュアル操作で変数の内容を呼び出し、予定した値になっているかどうかチェックできます。
このとき [▲] を押せば、押している間、止まっている行の内容を表示します。([▲] を押して離したときはプロンプト記号(>)が表示されますが、[▼] で続けて実行できます。)
[▼] を押したままにすると、連続的にトレースが実行されます。
● トレース中に、LOCATE命令で指定された画面の位置に結果などが表示された場合、次の行番号は、結果などが表示された行の次の行から表示されます。(LOCATEについては、BASICの各命令の説明を参照)
● LOCATE命令で表示開始位置が指定されているときに、マニュアル操作で変数の呼び出しや計算などを行うと、表示開始位置の指定は解除されます。
(注) トレースモードは、TROFF [↵] と押すか、[SHIFT] + [CA] (または [2ndF] [CA]) と押す、または電源を切るか、オートパワーオフ(自動節電機能)で電源が切れるまで設定状態が保持されます。

(2) プログラムの途中で実行を停止させてチェックする場合

トレースモードを設定しない場合でも、プログラムの実行を停止させたい位置にSTOP命令を書いておけば、STOP命令を実行した時点でブレークメッセージ(BREAK　IN　行番号)を表示して、プログラムの実行が停止します。
このとき、
①マニュアル操作で変数の内容をチェックする。
②続いて [▼] の操作で、以降の行を1行ずつトレースする。
などの操作でデバッグを行います。その後、通常の実行状態に戻すときは
　CONT [↵]
と押します。
● 通常のプログラム実行中に [ON] (BREAK) を押すと、現在実行している行の終わりで実行を停止し、ブレークメッセージを表示します。このときも前記と同様の操作を行うことができます。
なお、ブレーク状態(停止状態)のときに [▲] を押せば、押している間停止しているプログラム行が表示されます。

7. プログラムのファイル

本機は、プログラムをファイルとして内部メモリに保存しておくことができます。
メモリの一部をプログラム・データエリアと切り離して、プログラムファイルエリアとして確保し、このエリアにプログラムを保存します。
本機で実行できるプログラムは、プログラム・データエリアに入っているプログラムです。プログラムファイルエリアのプログラムを実行したいときは、プログラムファイルエリアからプログラム・データエリアに呼び出してから実行します。
ここでは、BASICプログラムのファイルのしかたについて説明します。
(注) ● BASICプログラムだけではなく、テキストのファイルもあります。TEXTモードの説明も参照してください。
● プログラムを記憶するプログラムファイルエリアと、データを記憶するラムデータファイルエリア(シーケンシャルデータ)があります。これらはそれぞれ独立して確保されます。なお、ラムデータファイルエリアについては、次項の「8. データのファイル」を参照してください。

(1)　ファイルに関する命令

ファイルに関する命令を簡単に説明します。くわしくは、ＢＡＳＩＣの各命令の説明を参照してください。

〈命　令〉	〈機　能〉
FILES	登録されているファイルのファイル名を表示します。
LFILES	登録されているファイルのファイル名をプリンタで印字します。
KILL	ファイルを消去します。
LOAD	ＢＡＳＩＣプログラムをプログラムファイルエリアからプログラム・データエリアに呼び出します。
SAVE	プログラム・データエリア内のＢＡＳＩＣプログラムをプログラムファイルエリアに保存（登録）します。

●ファイル名

ＫＩＬＬ、ＬＯＡＤ、ＳＡＶＥ命令を実行するときはファイル名を指定する必要があります。ファイル名はファイルの見出しのようなものです。

ファイル名は最大８文字で構成され、次の文字が使用できます。

　Ａ～Ｚ、ａ～ｚ、０～９、＃、＄、％、＆、’、（、）、－、＠、｛、｝、＿、°、カタカナ

また、ファイル名には拡張子をつけることができます。拡張子はファイルの種類を区別するためなどに利用します。拡張子はピリオドと３文字以内の文字で構成され、ファイル名の直後につけます。使用できる文字は、ファイル名と同じものです。（スペースは使用できません。）

なお、拡張子を省略すると自動的に「．ＢＡＳ」が指定されます。

●ファイル名の完全な記述は次の形式になります。

　〃ファイル名．拡張子〃

(2)　プログラムの登録（保存）（ＳＡＶＥ命令）

プログラム作成後、そのプログラムを登録する場合は、ＲＵＮモードまたはＰＲＯモードで次の操作を行います。

　〈例〉　ＳＡＶＥ〃ＴＥＳＴ〃 ↵

　　　　　　　└ファイル名

ファイルとして登録する場合、必ずファイル名が必要です。ファイル名には、拡張子をつけることができます。

この例では拡張子を省略していますが、省略すると“．ＢＡＳ”が自動的につけられます。

なお、プログラムファイルエリアはＳＡＶＥ命令でプログラムを登録したとき、自動的に必要な大きさが確保されます。ただし、メモリの残りが少なく、必要な大きさが確保できないときはエラー60になります。

(3)　ファイルの登録の確認（ＦＩＬＥＳ、ＬＦＩＬＥＳ命令）

プログラムを登録した場合、ＦＩＬＥＳまたはＬＦＩＬＥＳ命令で確認ができます。

　〈例〉　ＦＩＬＥＳ ↵

と操作すれば、登録されているファイル名が画面に表示されます。

登録されているファイルが多い場合は ▼ を押していくことにより、画面に呼び出すことができます。戻すときは ▲ を押します。

　〈例〉　ＬＦＩＬＥＳ ↵

と操作すれば、登録されているファイル名が印字されます。（ただし、プリンタＣＥ-126Ｐが接続されているときのみ有効です。）

(4)　プログラムの呼び出し（ＬＯＡＤ命令）

登録したプログラムは次の操作で呼び出すことができます。

　〈例〉　ＬＯＡＤ〃ＴＥＳＴ〃 ↵

この場合、ファイル名が“ＴＥＳＴ．ＢＡＳ”のファイル内容（プログラム）が呼び出されます。

登録したときに拡張子をつけたときは、拡張子まで完全に入力してください。ただし、拡張子が“．ＢＡＳ”の場合は拡張子を省略できます。

ＦＩＬＥＳ命令で呼び出した後、ファイル名を選んで呼び出すこともできます。

〈例〉		
	ＦＩＬＥＳ ↵	ファイル名を表示させます。
	▼…▼ ▲…▲	画面のファイル名の前にある矢印（➡）を、呼び出したいファイル名に移します。
	SHIFT ＋ M (LOAD)	矢印（➡）で示したファイルの内容（プログラム）が、プログラム・データエリアに呼び出されます。

ご注意

拡張子をつけて登録する場合、拡張子には“．ＴＸＴ”を使用しないでください。“．ＴＸＴ”を使用すると、テキストプログラムのファイルと区別がつかなくなります。（177ページ参照）

(5)　記録されているファイルの消去（ＫＩＬＬ命令）

ファイルを消去するときは、次の操作を行います。

　〈例〉　ＫＩＬＬ〃ＴＥＳＴ．ＢＡＳ〃 ↵

この場合、“ＴＥＳＴ．ＢＡＳ”というファイルが消去されます。

ＫＩＬＬ命令では、記録されているファイル名のとおりに、ファイル名を拡張子まで入力してください。

ただし、拡張子が“．ＢＡＳ”の場合は拡張子を省略できます。

なお、ＴＥＸＴモードで消去する方法もあります。（第５章を参照）

8．データのファイル

本機は、データをシーケンシャルファイルとして内部メモリに保存しておくことができます。この領域をラムデータファイルエリアと呼びます。

ラムデータファイルエリアは、あらかじめＩＮＩＴ命令でファイルの大きさを確保しておかなければなりません。**ラムデータファイルの使いかたについては、「第５章　２．９　ラムデータファイル」を参照してください。**

ラムデータファイルのデータはプログラム・データエリア内のプログラム（BASICプログラム）で、入出力（書き込みや読み出し）ができます。また、テキストエリアとの間でファイルの相互転送もできます。

9．別のポケコンへの記録、読み込み

EA-129C（ポケコン接続ケーブル）ともう1台のPC-G850VSをお持ちの場合は、プログラムを別のポケコンに記録したり、読み込んだりできます。ここでは、プログラムを別のポケコンに記録する、あるいは別のポケコンから読み込む場合の操作方法について説明します。

(1)　ポケコン間通信に関する命令

次にポケコン間通信に関する命令を簡単に述べます。くわしくは個々の命令の説明を参照してください。

〈命　令〉	〈機　　能〉
BLOAD	別のポケコンに記録されているBASICプログラムを読み込みます。
BLOAD ?	別のポケコンに記録されているBASICプログラムと計算機内のBASICプログラムの照合を行います。
BLOAD M	別のポケコンに記録されている機械語プログラムを読み込みます。
BSAVE	計算機内のBASICプログラムを別のポケコンに記録します。
BSAVE M	計算機内の機械語プログラムを別のポケコンに記録します。

(2)　準　備

本機、EA-129C、もう1台のポケコンを、それぞれを接続してください。接続するときは両方のポケコンとも電源を切ってから行ってください。

なお、EA-129Cは両方のポケコンの左の周辺機器接続端子（11ピン）に接続します。

(3)　別のポケコンへの記録方法

①本機にBASICプログラムを入れます。

②別のポケコンで読み込み命令を入力し実行します。

　〈例〉　RUNモード（またはPROモード）を指定

　　　　BLOAD ↵

③本機で記録命令を入力し実行します。

　〈例〉　RUNモード（またはPROモード）を指定

　　　　BSAVE ↵

● 記録が始まると別のポケコン画面の下の行の右端桁に＊マークが表示されます。

④記録が終了すると、プロンプト記号が表示されます。続いて次項の方法で“照合”を行ってください。

(4)　本機内と別のポケコンのBASICプログラムの照合

BASICプログラムを別のポケコンに記録した後、まちがいなく記録されたかどうか照合して確認します。

①本機で照合命令を入力し実行します。

　〈例〉　RUNモード（またはPROモード）を指定

　　　　BLOAD ? ↵

②別のポケコンで記録命令を入力し実行します。

　〈例〉　RUNモード（またはPROモード）を指定

　　　　BSAVE ↵

● BASICプログラムが見つかり、照合が始まると本機画面の下の行の右端桁に＊マークが表示されます。

③両方の内容がすべて一致していれば実行を終了し、プロンプト記号が表示されます。

　もし、エラー82になった場合は、もう一度最初から照合を行ってください。

　それでもなお、エラーになる場合は、もう一度“記録”から行ってください。

(5)　別のポケコンからの読み込み

BASICプログラムを別のポケコンから本機に読み込む場合は、照合命令を読み込み命令にかえて、照合の場合と同じ手順で行ってください。

　読み込み命令：BLOAD

　〈例〉　BLOAD ↵

もし、読み込み途中でエラー80になった場合は、もう一度最初から読み込みを行ってください。

10．プログラムの実行開始方法とラベルについて

(1)　プログラムの実行開始方法

プログラムの実行開始方法には次の方法があります。

RUN命令によるもの

RUN ↵ ………………プログラムの先頭から実行開始

RUN　行番号 ↵ ……指定した行番号から実行開始

RUN　ラベル ↵ ……指定したラベルが書かれている行から実行開始

GOTO命令によるもの

GOTO 行番号↵…指定した行番号から実行開始

GOTO ラベル↵…指定したラベルが書かれている行から実行開始

ラベルは、" "(ダブルクォーテーション)で前後を囲うか、*(アスタリスク)を前につけて指定します。(くわしくは、次の「ラベルについて」の項参照)

〈例〉 RUN"AB"
　　　GOTO*AB

これらの開始方法により、状態の解除や変数の消去などに違いがあります。

RUN命令による実行	GOTO命令による実行
・ウエイト(WAIT)指定 → 0に設定	・ウエイト(WAIT)指定 → 保持
・表示フォーマット(USING)指定 → 解除	・表示フォーマット(USING)指定 → 保持
・配列変数(DIM指定)、単純変数 → 消去	・配列変数(DIM指定)、単純変数 → 保持
・READ文に対するDATA → 初期化する	・READ文に対するDATA → 初期化しない
・LINE、GCURSORの位置指定 → 初期化する	・LINE、GCURSORの位置指定 → 初期化しない
・PRINT=LPRINTの指定 → 解除	・PRINT=LPRINTの指定 → 保持
・すべてのファイル → 閉じる	・すべてのファイル → 閉じない
・パラレルポート → 閉じる	・パラレルポート → 閉じない
・固定変数の内容 → 保持	・固定変数の内容 → 保持
・トレースモード(TRON)設定 → 保持	・トレースモード(TRON)設定 → 保持

(注) プログラムをRUN命令で実行したとき、配列変数や単純変数は消去されます。データを残しておきたいときは、GOTO命令で実行を開始してください。

(2) ラベルについて

次の例のように、ラベルはプログラムの行の先頭(行番号の次)に見出しとして書いておくものです。GOTOやGOSUB、THENなどのジャンプ先としてラベルを指定すれば、実行時に指定したラベルを探して、そこへジャンプします。

```
〈例〉 10  "AB":INPUT …
            └ラベルAB
       ⋮
       50  *CDE:CLEAR …
            └ラベルCDE
       ⋮
       100 IF … THEN"AB"  ←ラベルABへジャンプ
       ⋮
       150 GOTO *CDE      ←ラベルCDEへジャンプ
```

また、プログラム実行開始命令であるRUNやGOTO命令で指定すれば、指定したラベルのある行からプログラムを実行できます。たとえば、使用できる行番号の範囲内で複数のプログラムの先頭にラベルをつけて本機に書き込んでおき、ラベルを指定して実行を開始すれば、必要なプログラムを実行できます。

" "で囲んだラベルおよびラベル指定で使用できる文字

" "で囲んだラベルには、英字、数字、カタカナ、記号などが使用できます。

〈例〉 "ABCDE"
　　　"X10"
　　　"サブ ルーチン"

*を用いるラベルおよびラベル指定で使用できる文字

*記号を先頭につけるラベルには、英字か、英字とそれに続く数字が使用できます。

〈例〉 *START
　　　*S123

- ラベルは必ず英文字で始まっていなければなりません。
- 英字の小文字は大文字に変換されます。
- 予約語のスペルで始まる文字列は使用できません。
- カタカナや記号は使用できません。

本機では" "で囲んだラベルと、*を用いるラベルを混合して使うことができますが、1つのプログラムの中では、どちらかに統一されることをお勧めします。
なお、*を用いるラベルは多くのパソコンでも用いられています。

n進演算機能

n進演算機能では、10進、2進、16進について、それぞれの変換(基数変換)、補数変換、および計算ができます。(符号付き15ビットの演算)
n進演算機能は、ROM内に格納しているn進演算のBASICプログラムをプログラム・データエリアに呼び出して使用します。
呼び出したプログラムはPROモードで変更することもできます。

【プログラムのスタート】

RUNモードで SHIFT + BASE-n と押してください。
プログラムが呼び出され、同時にプログラムがスタートして、右の画面(**初期画面**)になります。

```
***** n シン エンザ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

(1, 2, 3, 4)?
```

1 〜 4 を押して、それぞれの機能を選びます。

SHIFT + BASE-n と押したときに、BASICプログラムエリアにプログラムがある場合は、右のように表示されます。

```
BASIC DELETE OK? (Y)
```

ｎ進演算を実行したいときは、ＢＡＳＩＣプログラムを消去しなければなりません。

ＢＡＳＩＣプログラムを消去するときは［Y］を押します。164ページの初期画面になります。
消したくないときは［CLS］など［Y］以外のキーを押してください。ＲＵＮモードになります。

（注）● ｎ進演算プログラムは、2794バイトの大きさがありますので、空きエリアが2794バイト以上ないと実行できません。また、実行時に変数として167バイト使用します。
プログラムを消去しても空きエリア（2794バイト必要）が不足するようなときは、前記操作で［Y］を押しても、プログラムの消去を行わずにＲＵＮモードになります。
プログラムは存在しないが空きエリアが足りない（データが多い）ときも、［SHIFT］＋［BASE-n］と押しても何も実行せずにＲＵＮモードになります。
なお、実行時に変数が確保できない（167バイト必要）とエラー60になります。
このような場合は、変数を消去したり、ＴＥＸＴプログラムを消去したりして、2961（2794＋167）バイト以上空けてください。
● ［SHIFT］＋［BASE-n］と押すと、ＷＡＩＴ指定、ＵＳＩＮＧ指定、トレースモード（ＴＲＯＮ）は解除されます。

ROMについて

ＲＯＭとは Read Only Memory のことで、読み出し専用のメモリです。関数やＢＡＳＩＣなどのソフトウェアが格納されているところです。

プログラムの実行を中止するときは、［BREAK］を押します。再度、最初からスタートするときはＲＵＮ命令で実行を開始してください。［SHIFT］＋［BASE-n］と操作すると、呼び出されているｎ進演算プログラムを消去して、再度呼び出すことになります。
なお、ｎ進演算のプログラムは、約５分間キー操作をしないと自動的に実行を中止して、プロンプト記号（>）を表示します。

【プログラムの使いかた】

１．概要

初期画面では、［1］～［4］で、入力、変換、補数、計算の４つの機能を選ぶことができます。

１：入力……変換や計算の元になる数値を入れます。
２：変換……表示している数値を次の順番で変換します。また、何進数（10進、16進、２進のいずれか）で入力や計算を行うかも指定します。
　　10進数 → 16進数 → ２進数 → 10進数 …
３：補数……表示している数値の結果の補数を求めます。
４：計算……10進、16進、２進の四則計算（＋－＊／）と、論理計算（ＡＮＤ、ＯＲ、ＮＯＴ、ＸＯＲ）ができます。

このプログラムで扱える数値はそれぞれ、次のようになります。

２進……16桁以内の２進数値（16ビット目は符号）
　17桁以上入力したときは、先頭から16桁が有効になります。また、計算結果が16桁を超える場合は、超えた分、上位桁が切り捨てられます。
16進……０～ＦＦＦＦの16進数値（８０００～ＦＦＦＦは負数）
　５桁以上の16進数値を入力した場合は、後ろ４桁が有効になります。計算結果が４桁を超える場合は、超えた分、上位桁が切り捨てられます。
10進……－32768～32767の10進数値
　10進数は、内部で16進数に変換されて処理されます。したがって、16進数に変換されたときに、上記の16進数の範囲になる値を扱います。

［補足］
・変換結果などを表示しているときに［1］を押すと、表示している進数での入力画面になります。
・入力した内容が、そのときに処理または計算できない値や数字、文字、桁数のときは、ＥＲＲＯＲを表示した後、入力待ちの状態に戻ります。
・16進数のＡ～Ｆは英文字のキーで入力します。

２．実行例

プログラムを実行させて、初期画面にしてください。

〈例１〉
10進数の1230を16進数および２進数に変換します。
“入力”を選びます。
［1］

```
***** n シン エンザ゛ン *****
  1：ニュウリョク    2：ヘンカン
  3：ホスウ          4：ケイサン

[1 0 シン] =_
```

10進数の入力待ちになります。
（初期画面のときは10進数のモードになっています。）

1230を入力します。
１２３０［↵］

```
***** n シン エンザ゛ン *****
  1：ニュウリョク    2：ヘンカン
  3：ホスウ          4：ケイサン

[1 0 シン] =                 1 2 3 0
```

“変換”を選びます。
［2］

```
***** n シン エンザ゛ン *****
  1：ニュウリョク    2：ヘンカン
  3：ホスウ          4：ケイサン

[1 6 シン] =                 0 4 C E
```

16進数に変換されます。

もう一度“変換”を選びます。
［2］

```
***** n シン エンザ゛ン *****
  1：ニュウリョク    2：ヘンカン
  3：ホスウ          4：ケイサン

[  2 シン] =  0 0 0 0 0 1 0 0 1 1 0 0 1 1 1 0
```

２進数に変換されます。

もう一度“変換”を選ぶと10進数に戻ります。

〈例２〉
16進数１２Ｃ７の補数を求めます。
“変換”により、16進を選びます。
［2］

```
***** n シン エンザ゛ン *****
  1：ニュウリョク    2：ヘンカン
  3：ホスウ          4：ケイサン

[1 6 シン] =                 0 4 C E
```

１２Ｃ７を入力します。
　1（入力を選びます）
　１２Ｃ７↵

1を押して“入力”を選ぶことを忘れないようにしてください。

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン] =                   12C7
```

補数を求めます。
　3

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン] =                   ED39
```

もう一度、3を押して補数変換をすれば、ＥＤ３９が１２Ｃ７に戻ります。（ＥＤ３９と１２Ｃ７はお互いに補数の関係です。）

参　考

ここでいう補数は、２進数においては２の補数、16進数においては16の補数です（基数に対する補数）。10進数においては、符号を反転させて補数としています。
この場合の補数の関係は、互いを加えると０になる関係です（ただし、最上位桁に生じる桁上げは無視するものとする）。
たとえば、１２Ｃ７ＨとＥＤ３９Ｈを加えると１００００Ｈになります。このとき、最上位桁の１を無視すると００００Ｈになります。（Ｈは16進数を示す記号）

〈例３〉

16進数３Ｅ７Ｃと０ＦＦ０のＡＮＤ（論理積）を求めます。
16進を選び、３Ｅ７Ｃを入力します。
　2（16進にします）
　1 ３Ｅ７Ｃ↵

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン] =                   3E7C
```

“計算”を選びます。
　4

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン]                     3E7C
(+, -, *, /, A, O, N, X) ?
```

＋、－、＊、／……加算、減算、乗算、除算
Ａ……ＡＮＤ　　Ｏ……ＯＲ
Ｎ……ＮＯＴ　　Ｘ……ＸＯＲ

計算命令（ＡＮＤ）を選び、０ＦＦ０を入力します。
　A
　０ＦＦ０

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン]                     3E7C
AND0FF0_
```


実行します。
　↵

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン] =                   0E70
```

０Ｅ７０が求まります。

〈例４〉

16進数１Ｅ０１から10進数の96を引き、結果を16進数で求めます。
１Ｅ０１を入力して、10進数に変換します。
　2（16進にします）
　1 １Ｅ０１↵
　2 2

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[10シン] =                   7681
```

計算をします。
　－ ９６↵
“計算”を選ぶ4の操作は省略しています。

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[10シン] =                   7585
```

結果を16進数に変換します。
　2

```
***** n シン エンサ゛ン *****
  1:ニュウリョク    2:ヘンカン
  3:ホスウ          4:ケイサン

[16シン] =                   1DA1
```

参　考

次のファイル名でプログラムをプログラムファイルエリアに登録しておけば、SHIFT＋BASE-nの操作で、呼び出し、実行ができます。
　“ＢＡＳＥ_Ｎ．ＢＡＳ”　　（注）_はアンダーラインです。
プログラムファイルエリアにこのファイル名のプログラムがあると、内部のｎ進演算プログラムは呼び出されません。
したがって、ｎ進演算プログラムを変更して使用するときは、このファイル名で変更したプログラムを登録しておけば、元のプログラムと同じようにSHIFT＋BASE-nと押して呼び出し、実行することができます。（ただし、元のプログラムを呼び出すときは、このファイルを消去する必要があります。）

なお、SHIFT＋BASE-nと押した場合、自動的に　ＧＯＴＯ１００　を実行してプログラムをスタートさせます。したがって、プログラムは100行から書いてください。

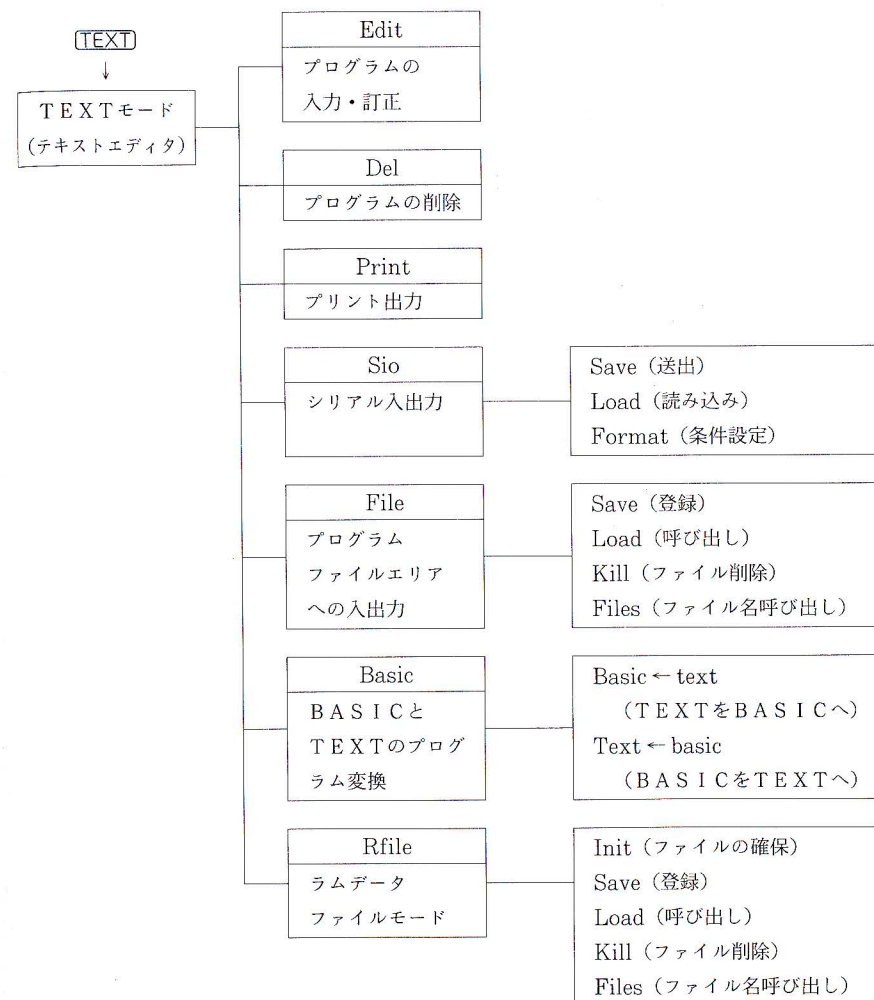
第5章
TEXTモード（テキストエディタ）

TEXTモード（テキストエディタ）では、アスキー形式でのプログラムの入力、編集、SIOへの入出力やラムデータファイル（データファイル）の確保などが行えます。
本機のBASICの各命令は、中間コードと呼ばれる2バイトコードに変換して記憶しています。このコードはハードウェアやBASICインタプリタにより異なるため、そのままではパソコン等との通信はできません。
アスキーコードは、アルファベットや数字、基本的な記号がほとんどの機種で共通であるため、パソコン等ではアスキーコードでの通信が多く行われています。
本機では、パソコン等との通信を容易に行えるよう、アスキー形式でプログラムを作成・編集したり、保存したり、中間コード形式（BASIC）とアスキー形式（TEXT）の相互変換をしたりできるTEXTモードを設けています。ラムデータファイルのデータもアスキー形式のデータに変換できます。
本章では、このTEXTモードの個々の機能について説明します。

テキストエディタで書き込んだ内容は次の操作でコンパイルもしくはアセンブルします。

- C言語のとき
  SHIFT ＋ TEXT(C) を押してから C (Compile) を押します。
- CASLのとき
  SHIFT ＋ ASMBL を押してから C (Casl) を選んだ後、A (Assemble) を押します。
- 機械語プログラム（Z80アセンブラのソースプログラム）のとき
  SHIFT ＋ ASMBL を押してから A (Assembler) を選んだ後、A (Asm) を押します。
- PICのとき
  SHIFT ＋ ASMBL を押してから P (Pic) を選んだ後、A (Assembler) を押します。

1. TEXTモード機能一覧

下記のTEXTモードの機能概略図を参考にしてください。
なお、C言語、CASL、Z80アセンブラおよびPICソースプログラムの入力・編集も、TEXTモードで行います。

2. TEXTモードの使いかた

2. 1 TEXTモードの設定

RUNモードやPROモードなどで TEXT を押すと、右の画面になります。
この画面を**機能選択画面**と呼びます。
この画面で、表示されている各機能をその頭文字（大文字）に相当するキーを押して選びます。

```
*** TEXT EDITOR ***

Edit Del  Print
Sio  File Basic Rfile
```

各機能を選ぶと、それぞれの機能のメニュー画面になったり、機能が働いたりします。

- 各機能が働いている状態で、動作を止めたり、メニュー画面に戻したり、機能選択画面に戻したりするときは、BREAK を押します。
  ただし、エラー状態を解除したり、ファイル名など入力中の文字などを消去するときは CLS を押してください。
- TEXTモードは、BASIC 、SHIFT + ASMBL などのキーでモードを変えたり、電源を切ると解除されます。

2. 2 エディット機能（Edit）

機能選択画面で E を押せば、エディット機能が選ばれ、エディット画面になります。

E

```
TEXT EDITOR
<
```

- エディット機能では、プロンプト記号が“＜”になります。（BASICモードでは“＞”）
- TEXTプログラムを書き込むときは、BASICプログラムの場合と同じように、行番号を先頭につけて書き込みます。ただし、BASICのように、自動的に行番号の後ろにコロン(:)をつけたり、命令の後ろにスペースを入れたりはしません。入力したとおりに書き込まれます。
- 行の順番は、行番号の順番に並べ替えられます。
- 行番号は、1～65279の範囲でつけることができます。この範囲を超えている場合、または行番号がない場合はエラー（LINE NO. ERROR）になります。エラーは CLS で解除します。
- 機能選択画面に戻るときは BREAK を押します。

（注）行番号の次が数字で始まるようなテキスト行を入力することはできません。
数字で始まるような行を入力するときは ’（シングルクォーテーション）で行番号と数字を区切って入力してください。

〈例〉 50’100␣FORMAT（17X, A）↵
行番号 ↑ シングルクォーテーション

〈例〉 次のプログラムを入力します。

```
10INPUT A
20B=A*A
30PRINT A, B
40END
```

［キー操作］

10INPUT SPACE A ↵
20B=A*A ↵
30PRINT SPACE A, B ↵
40END ↵

```
10INPUT A
20B=A*A
30PRINT A, B
40END
```

ご注意
このプログラムはパソコン等へ転送することを考えたプログラムで、CASLのプログラムではありません。そのため、このプログラムをCASLでアセンブルするとエラーになります。CASLの文法については、CASLの説明を参照してください。

プログラムの編集

TEXTプログラムの変更、修正などはBASICプログラムの変更、修正と同じ方法で行うことができますので参照してください。
なお、次のようにBASIC命令と同等機能の命令が用意されています。それぞれの命令の働きについては、BASICのそれぞれの命令も参照してください。

AUTO命令…………A命令（オート）
LIST命令…………L命令（リスト）
RENUM命令………R命令（リナンバー）
DELETE命令……D命令（デリート）
LCOPY命令………C命令（コピー）

さらに、TEXTモードでは検索（S命令）・置換（E命令）命令が用意されていますので、編集が容易です。
ただし、TEXTモードでのリナンバーは、行頭の番号だけをつけ直します。
BASICプログラムをTEXTプログラムに変換して、リナンバーを行うとGOTOやTHEN、GOSUB、RESTORE命令などの後の行番号は変更されません。再びBASICプログラムに変換したとき、正しく動作しなくなりますので注意してください。

A命令書式 A ［[開始行] ［, 増分]] ↵
開始行で指定した行番号から、それ以降の行番号を増分で指定した値に従って自動発生します。

L命令書式 (1) L ↵
(2) L行番号 ↵
(3) Lラベル ↵

R命令書式 R ［新行番号］［, ［開始行］［, 増分]] ↵
開始行で指定したプログラムの行番号を、新行番号に書き換え、それ以降の行番号を増分

で指定した値に従って順次書き換えていきます。

D命令書式　(1)　D 開始行番号［,［終了行番号］］↵

開始行番号から終了行番号までのプログラムを削除します。終了行番号を省略したときは最終行まで削除します。コンマ（，）も含めて省略したときは開始行番号だけを削除します。

(2)　D ，終了行番号

最初の行から終了行番号までを削除します。

C命令書式　C　コピー元開始行番号，コピー元終了行番号，コピー先開始行番号 ↵

コピー元開始行番号からコピー元終了行番号までのプログラム行を、コピー先開始行番号からコピーします。コピー先プログラム行の増分は、コピー元プログラム行の増分と同じです。

S命令書式　S［式,］"文字列" ↵

指定した文字列を検索し、停止します。文字列は16文字まで指定できます。↵ を押すと次の文字列の検索を行い、CLS を押すと検索を終了します。また、最終行（または最初の行）まで検索を行っても終了します。

式で検索の方向を指定します。

1：最初の行より行番号の大きい方向に検索します。

0：最終行より行番号の小さい方向に検索します。

省略した場合は“1”の指定になります。

(注)・行番号は検索できません。

・"（ダブルクォーテーション）を検索するときは、文字列として¥"を指定し、¥を検索するときは¥¥を指定します。　例：S"¥""

E命令書式　E［式,］"文字列1"，"文字列2" ↵

文字列1を検索し、停止します。文字列1、文字列2は16文字まで指定できます。

↵ を押すと文字列1を文字列2に置換（置き換え）します。

SPACE を押すと置換を行わず次の文字列の検索・置換に進みます。

CLS を押すと検索・置換を終了します。

式で検索の方向を指定します。

1：最初の行より行番号の大きい方向に検索します。

0：最終行より行番号の小さい方向に検索します。

省略した場合は“1”の指定になります。

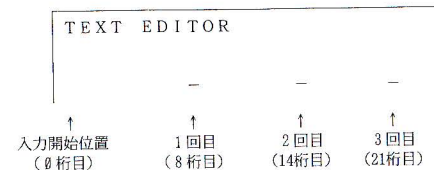
(注)・置換することにより1行が255文字を超えると、"STRING TOO LONG" のエラーが表示されます。

・行番号は検索できません。

・"（ダブルクォーテーション）を検索するときは、文字列として¥"を指定し、¥を検索するときは¥¥を指定します。　例：E0，"¥¥"，"A"

（［ ］内の指定は省略可です。）

TAB の働き

エディット機能の中での TAB は、次の図に示すようにカーソルを送ります。

最初 TAB を押すとカーソルが8桁進み、2回目を押すと6桁進みます。3回目以降は7桁ずつ進みます。

2．3　TEXTプログラムの消去（Delete）

機能選択画面で D を押せば、消去確認画面になります。

D

Y を押せば、TEXTプログラムなど、テキストエリアの内容が消去され、機能選択画面に戻ります。

Y 以外のキーを押すと、消去されずに機能選択画面に戻ります。

● TEXTモードでの記憶内容がない場合は、機能選択画面で D を押しても何も実行されません。

2．4　TEXTプログラムの印字（Print）

プリンタCE-126Pをお持ちの場合は、接続して電源を入れ、機能選択画面で P を押せば印字します。

P

*** TEXT EDITOR ***

--- PRINTING ---

印字が終われば、機能選択画面に戻ります。

● 印字を途中で止めるときは BREAK を押してください。

● プリンタの電源が入っていないときや、プリンタが接続されていないときは、機能選択画面で P を押しても何も実行されません。

2．5　シリアル入出力（Sio）

機能選択画面で S を押せば、シリアル入出力のメニュー画面になります。

S

この画面で、出力（Save）、読み込み（Load）、条件設定（Format）を選びます。それぞれの機能の頭文字に相当するキーを押します。

● 通信を行う前に、入出力条件を通信する相手側と合わせておく必要があります。

①入出力の条件設定（Format）

通信を行う場合の入出力の条件を設定します。

シリアル入出力のメニュー画面で[F]を押すと、次の画面になります。しばらく待つか、いずれかのキーを押すと、条件設定画面になります。

[F]

```
        << SIO >>

 Select   ←, →, ↑, ↓ key
 Set      ↵ key

  --- push any key ---
```

```
→baud rate   =1200
 data bit    =8
 stop bit    =1
 parity      =none
 end of line =CR LF
 end of file =1A
```

（条件設定画面）

➡マークは変更できる項目を示しています。変更したい項目を[▼]、[▲]で移動させて選択します。条件は8種類あり、[▼]を押していくと、隠れている部分が1行ずつ表示されます。

⋮
[▼]

```
 stop bit    =1
 parity      =none
 end of line =CR LF
 end of file =1A
 line number =yes
→flow        =RS/CS
```

初期状態では以上の値が設定されています。

項目を選択し、[▶]または[◀]で条件（値）を切り替えます。ただし、end of file（エンド　オブ　ファイル）は直接コードを入力します。
そして[↵]を押せば設定されます。[↵]を押さないと、前の条件のままになります。

【条件の説明】

- **通信速度**（baud rate）：300、600、1200、2400、4800、9600
  データ転送速度の指定で、本機では上記の速度（bps）が指定できます。
  bps：bit per second（1bps＝1ビット／秒）
- **ワード長**※（data bit）：7、8
  1文字を何ビット構成で送受信するかを指定します。
- **ストップビット数**※（stop bit）：1、2
  ストップビット数を1ビットにするか2ビットにするかを指定します。
- **パリティ**（parity）：none、even、odd
  パリティビットをどのように扱うかを指定します。
  none …パリティビットの送受信を行いません。
  even …偶数パリティが指定されます。
  odd……奇数パリティが指定されます。
- **区切りコード**（end of line）：CR、LF、CR　LF
  プログラムライン（行）の終了を示す区切りコードを指定します。
  CR……CR（キャリッジリターン）コードが指定されます。
  LF……LF（ラインフィード）コードが指定されます。

  CR　LF…CRコード＋LFコードが指定されます。
  (注)ここで指定するコードはテキストプログラムの行末を示すコードです。
  PRINT＃やfprintfで出力されるデータの区切りコードは、常にCR　LFが指定されます。
- **テキスト終了コード**（end of file）：00～FF（2桁の16進数値）
  プログラムなどの終了を示すテキスト終了コードを指定します。
- **行番号つき入出力**（line number）：yes、no
  - **出力時**
    yes……行番号をつけて出力します。
    no……行番号を省いて出力します。
  - **読み込み時**
    yes……行番号を付加しません。
    読み込むプログラムに行番号がついているときに指定します。
    読み込んだプログラムに行番号がついていないと、エラー（LINE NO. ERROR）になります。
    no……自動的に行番号（10から10きざみ）をつけます。
- **フロー制御**※（flow）：RS／CS、Xon／Xoff、none
  通信を行うときの制御方法を指定します。
  RS／CS……RS／CS信号で制御します。
  Xon／Xoff……Xon／Xoffコードで制御します。
  none……制御を行いません。

※OPEN"COM1:"およびfopen("stdaux1",として全二重通信を指定しているときは、常にワード長は8ビット、ストップビット数は1ビットになります。これ以外の場合は、常にフロー制御はRS/CSに固定されます。
入出力条件は前に述べた方法で変更できます。変更した入出力条件はリセットスイッチを押してメモリを消去するか、電池交換を行うか、条件の変更を行うまで保持されます。

②出力（Save）

シリアル入出力のメニュー画面で、[S]を押すとシリアル入出力端子への出力が開始されます。

[S]

```
   << SIO >>

 --- SENDING ---
```

出力が終了すれば、シリアル入出力のメニュー画面に戻ります。
- 出力を中断するときは、[BREAK]を押します。メニュー画面に戻ります。
- TEXTモードでの記憶内容がない場合は、[S]を押しても何も行われません。

③読み込み（Load）

シリアル入出力のメニュー画面で、[L]を押すとシリアル入出力端子へ送られてくるデータの読み込みが開始されます。

[L]

```
<< SIO >>

--- RECEIVING ---
```

読み込みが終了（テキスト終了コードを受信）すれば、シリアル入出力のメニュー画面に戻ります。

- 読み込みを中断するときは[BREAK]を押します。メニュー画面に戻ります。
- データが正常に読み込めない場合や、パリティチェックで異常が発生した場合などではエラー（I／O DEVICE ERROR）になります。[CLS]でエラーを解除してください。

2．6 ミニI／OからのTEXTプログラム送出

TEXTモードの機能選択画面で[L]を押すと、ミニI／OのパラレルポートからTEXTプログラムを送出します。
詳しくは、先生の指導に従ってください。(366ページを参照)

2．7 プログラムファイル（File）

機能選択画面で[F]を押すと、プログラムファイルのメニュー画面になります。

[F]

```
<< PROGRAM FILE >>
Save   Load   Kill   Files
```

この画面で、登録（Save）、呼び出し（Load）、削除（Kill）およびファイル名の確認（Files）を選びます。それぞれの頭文字に相当するキーを押します。

- 登録したファイルはプログラムファイルエリアに書き込まれます。(158ページ参照)

①TEXTプログラムの登録（Save）

プログラムファイルのメニュー画面で[S]を押すと、ファイル名の入力待ちになります。

[S]

```
<< PROGRAM FILE >>
➡Save   Load   Kill   Files

FILE NAME=?
```

ファイル名を入力して[↵]を押せば、登録が行われます。

〈例〉ファイル名を“TEST”とします。

TEST

```
<< PROGRAM FILE >>
➡Save   Load   Kill   Files

FILE NAME=TEST_
```

[↵]

登録が行われ、プログラムファイルのメニュー画面に戻ります。
入力したファイル名がすでに登録されているときは、すでに登録されているファイルの内容を新しい内容に書きかえてよいかどうかを聞いてきます。（FILE OVERWRITE OK？（Y）と表示）
[Y]を押せばファイルが登録され、プログラムファイルのメニュー画面に戻ります。
[Y]以外のキー（そのとき有効に働くキー）を押せば、登録は中止されます。

- ファイル名を省略することはできません。ファイル名を入れずに[↵]を押すとエラー（ILLEGAL FILE NAME）になります。
- ファイル名は8文字以下の名前と、3文字以下の拡張子を指定できます。
  拡張子をつけなかった場合は、自動的に“．TXT”がつけられます。
- 空きエリアの容量が足りない場合は、エラー（MEMORY OVER）になります。
  空きエリアには、ファイルサイズ+34バイト（ファイル名の記憶などに使用）以上が必要です。
- テキストエリアにTEXTプログラムがない場合は、登録は行われません。

②TEXTプログラムの呼び出し（Load）

プログラムファイルのメニュー画面で[L]を押すと、登録されているプログラムファイル名を表示し、最初のファイル名の左側に“LOAD ➡”と表示されます。(何も登録されていないときは、画面は変わりません。)

[L]

（画面は、すでにこれらのプログラムが登録されている場合の例です。）

```
LOAD ➡ABC       .TXT    456
      PRO       .TXT   1234
      ｻﾝﾌﾟﾙ001.BAS   1567
      TEST      .TXT    789
```

“LOAD ➡”表示を[▼]、[▲]で呼び出したいファイル名の前に移し、[↵]を押せば、そのファイルの内容が呼び出され、メニュー画面に戻ります。表示されているファイル名以外にファイルがある場合は、[▼]、[▲]で移動させていくことにより1行ずつ表示されます。

- TEXTモードで呼び出せるプログラムは、TEXTモードで登録したプログラムだけです。BASICプログラム（BASICのSAVE命令で登録したプログラム）を呼び出そうとするとエラー（FILE MODE ERROR）になります。

③ファイルの削除（Kill）

プログラムファイルエリアに登録されているプログラム（ファイル）を削除します。
プログラムファイルのメニュー画面で[K]を押すと、ファイル名の入力待ちになります。(何も登録されていないときは、画面は変わりません。)

[K]

```
<< PROGRAM FILE >>
Save   Load   ➡Kill   Files
FILE NAME=?
```

削除したいプログラムのファイル名を入力して[↵]を押します。

〈例〉ファイル名“TEST”のプログラムを削除します。

TEST

```
Save   Load  ➜Kill   Files

FILE NAME=TEST_
```

[↵]

ファイルの削除確認画面になります。

（FILE　DELETE　OK?（Y）と表示されます。）

[Y]を押せばファイルが削除され、プログラムファイルのメニュー画面に戻ります。

[Y]以外のキー（そのとき有効に働くキー）を押せば、削除は中止されます。

● 拡張子の指定を省略した場合は、“.TXT”の指定とみなされます。

● 指定したファイル名が見つからない場合はエラー（FILE　NOT　FOUND）になります。

④ファイル名とファイルサイズの確認（Files）

プログラムファイルのメニュー画面で[F]を押すと、登録されているファイル名とファイルサイズが表示され、最初のファイル名の左側に ➜ マークが表示されます。（何も登録されていないときは、画面は変わりません。）

[F]

（画面は、すでにこれらのプログラムが登録されている場合の例です。）

```
➜ABC        .TXT     456
 PRO        .TXT    1234
 ｻﾝﾌﾟﾙ001.BAS    1567
 TEST       .TXT     789
```

表示されているファイル名以外にファイルがある場合は、[▼]、[▲]で ➜ マークを移動させていくことにより1行ずつ表示されます。

● このファイル名の確認画面で、[SHIFT]＋[M](LOAD)（または[2nd F][M](LOAD)）と操作すると、➜マークで示しているファイル名のプログラムを呼び出すことができます。

ファイルサイズについて

● テキストファイルのファイルサイズは、各行のバイト数の合計です。各行のバイト数は、3バイト（ラインナンバー）＋テキストのバイト数＋1バイト（改行）で計算されます。

〈例〉10␣INPUT␣A[↵]

ラインナンバー　テキスト　改行

この行のバイト数は、3＋8＋1＝12バイトです。

● RUNモードやPROGRAMモードで登録したBASICプログラムのファイルサイズは中間コードに変換後のサイズです。（RUNモードやPROGRAMモードで登録したBASICプログラムは中間コードに変換して登録されています。）

2．8　BASICコンバータ（Basic）

BASICプログラム（中間コード形式）をTEXTプログラム（アスキー形式）に変換したり、逆にTEXTプログラムをBASICプログラムに変換したりする機能です。本機用のBASICプログラムをパソコン等で管理する場合などに利用します。

機能選択画面で[B]を押すと、BASICコンバータのメニュー画面になります。

[B]

```
<< BASIC CONVERTER >>

Basic←text   Text←basic
```

この画面で、BASICへの変換（Basic ← text）、TEXTへの変換（Text ← basic）を選ぶことができます。それぞれの頭文字に相当するキーを押します。

①TEXTとBASICのプログラム変換（アスキー形式と中間コード形式の変換）

メニュー画面で、[B]を押せばTEXTプログラムをBASICプログラムに変換し、プログラム・データエリアに入れます。

[T]を押せばBASICプログラムをTEXTプログラムに変換し、TEXTエリアに入れます。

〈例〉TEXTプログラムをBASICプログラムに変換します。

[B]

```
<< BASIC CONVERTER >>

--- CONVERTING ---
```

変換を実行します。その後、機能選択画面に戻ります。

（変換する内容が少ない場合、この画面の表示は一瞬で終わります。）

● BASICプログラムに変換するとき、BASICのプログラム・データエリアにBASICプログラムがあった場合、あるいはTEXTプログラムに変換するとき、テキストエリアにTEXTプログラムがあった場合は、以前のプログラムを削除してよいかを聞いてきます。

[B]

```
➜Basic←text   Text←basic

BASIC DELETE OK? (Y)
```

[Y]を押せば、以前のプログラムが削除され、変換が開始されます。

[Y]以外のキー（そのとき有効に働くキー）を押せば変換は中止され、機能選択画面に戻ります。

● 通常、変換した場合でも変換元のプログラムは保持していますが、メモリ容量が足りない場合は、次のように、変換元のプログラムを削除してよいか聞いてきます。

```
--- CONVERTING ---
TEXT DELETE OK? (Y)
```

[Y]を押せば、変換を行いながら変換元のプログラムを消去していきます。変換が終われば、変換元のプログラムはすべて消去されます。

[Y]以外のキー（そのとき有効に働くキー）を押せば変換は中止され、機能選択画面に戻ります。

2．9　ラムデータファイル（Rfile）

シーケンシャルデータをラムデータファイルエリアに登録し、BASICプログラムなどで任意に読み出して使用することができます。

機能選択画面で[R]を押すと、ラムデータファイルのメニュー画面になります。

```
<< RAM DATA FILE >>

Init  Save  Load  Kill
Files
```

このメニュー画面で、ファイルの確保（Init)、テキストエリアからの登録（Save)、テキストエリアへの呼び出し（Load)、ファイルの削除（Kill）およびファイル名の確認（Files）を選びます。それぞれの頭文字に相当するキーを押します。

①ファイルの確保（Init）

ラムデータファイルを使用するときは、あらかじめInit機能でファイルの名前と大きさを確保しておかなければなりません。

ラムデータファイルのメニュー画面で[I]を押すと、ファイル名の入力待ちになります。

[I]

```
→Init  Save  Load  Kill
 Files

FILE NAME=?
```

〈例〉ファイル名を“TEST”とします。

TEST. DAT[↵]

```
→Init  Save  Load  Kill
 Files

FILE SIZE=?
```

次にファイルの大きさをバイト単位で指定します。このファイルには、指定した容量の範囲内でデータを登録することができます。

ここでは１０２４バイトの容量にします。

１０２４

```
→Init  Save  Load  Kill
 Files

FILE SIZE=1024_
```

[↵]

入力したファイル名がすでに登録されているときは、すでに登録されているファイルを消去して、新しいファイルとして確保してよいかどうかを聞いてきます。（FILE INITIALIZE OK?（Y）と表示されます。）

[Y]を押せば、新しいファイルを確保します。

[Y]以外のキー（そのとき有効に働くキー）を押せば、ラムデータファイルのメニュー画面に戻ります。

［補足］
- 拡張子を省略した場合は、自動的に“．DAT”がつけられます。
- ファイル名は８文字以内の文字で指定します。
- 指定した容量以上のデータは登録できませんので、少し大きめの容量を確保しておくことをおすすめします。ただし、あまり容量を大きくすると、フリーエリア（メモリの未使用部分）が少なくなります。なお、ファイルの最後に終了コードを書き込みますので、実際に書き込めるバイト数は（指定した容量−１）バイトです。
- ファイル名などを記憶する制御用として、34バイト使用します。指定する容量（バイト）＋34

バイト以上のフリーエリアがないと“MEMORY　OVER”エラーになります。なお、容量として０バイトを指定すると無視されます。

②テキストエリアからの登録（Save）

テキストエリアに書き込まれているプログラム（データ）を、データとしてラムデータファイルに登録します。登録するとき行番号は省略して登録します。

ラムデータファイルのメニュー画面で[S]を押すと、ファイル名の入力待ちになります。

[S]

```
<< RAM DATA FILE >>

Init →Save  Load  Kill
Files

FILE NAME=?
```

〈例〉ファイル名を“TEST”とします。

TEST

```
Init →Save  Load  Kill
Files

FILE NAME=TEST_
```

[↵]

入力したファイルにデータがすでに登録されているときは、すでに登録されているデータを新しいデータに書きかえてよいかどうかを聞いてきます。

（FILE　OVERWRITE　OK?（Y）と表示されます。）

[Y]を押せばファイルが登録され、ラムデータファイルのメニュー画面に戻ります。

[Y]以外のキー（そのとき有効に働くキー）を押せば、登録は中止されます。

- 登録が行われメニュー画面に戻ります。
- 拡張子の指定を省略した場合は、“．DAT”の指定とみなされます。
- 指定したファイル名がなかったり、容量が足りなかった場合は、エラーになります。
- TEXTモードでの記憶内容がない場合は、[↵]を押しても何も実行されずにメニュー画面に戻ります。
- ラムデータファイルへの登録（書き込み）はBASICプログラムのPRINT#命令で行うこともできます。

③テキストエリアへの呼び出し（Load）

ラムデータファイルに登録されているデータを、テキストエリアに呼び出しします。呼び出しするとき行番号は自動的に10番ごとに付加します。

ラムデータファイルのメニュー画面で[L]を押すと、登録されているファイル名と確保したサイズを表示し、最初のファイル名の左側に“LOAD →”と表示されます。（何も登録されていないときは、画面は変わりません。）

[L]

（画面は、すでにこれらのファイルが登録されている場合の例です。）

```
LOAD →TEST    .DAT  1024
      ABC     .DAT   512
      SAMPLE  .DAT  2048
```

“LOAD →”表示を[▼]、[▲]で呼び出ししたいファイル名の前に移し、[↵]を押すと、そのファイルの内容がテキストエリアに行番号付きで呼び出しされ、メニュー画面に戻ります。

表示されているファイル名以外にファイルがある場合は、[▼]、[▲]で移動させていくことにより１行ず

つ表示されます。

④ファイルの削除（Kill）

ラムデータファイルエリアに登録されているデータファイルを削除します。

ラムデータファイルのメニュー画面で K を押すと、ファイル名の入力待ちになります。(何も登録されていないときは、画面は変わりません。)

K

```
  << RAM DATA FILE >>

 Init  Save  Load ➜Kill
 Files

FILE NAME=?
```

削除したいデータファイルのファイル名を入力して ↵ を押すと、削除が行われます。

〈例〉ファイル名“TEST”のファイルを削除します。

TEST ↵

ファイルの削除確認画面になります。(FILE　DELETE　OK？（Y）と表示されます。)

Y を押せば削除が行われ、メニュー画面に戻ります。

Y 以外のキー（そのとき有効に働くキー）を押せば、削除は中止されます。

- 拡張子の指定を省略した場合は、“.DAT”の指定とみなされます。
- 指定したファイル名が見つからない場合はエラーになります。

⑤ファイル名と確保したサイズの確認（Files）

ラムデータファイルのメニュー画面で F を押すと、登録されているファイル名と確保したサイズを表示し、最初のファイル名の左側に ➜ マークが表示されます。(何も登録されていないときは、画面は変わりません。)

F

（画面は、すでにこれらのファイルが登録されている場合の例です。）

```
➜TEST    . DAT   1024
 ABC     . DAT    512
 SAMPLE  . DAT   2048
```

表示されているファイル名以外にファイルがある場合は、▼、▲ で ➜ マークを移動させていくことにより1行ずつ表示されます。

- このファイル名の確認画面で、SHIFT ＋ M（LOAD）（または 2nd F M（LOAD））と操作すると、➜ マークで示しているファイル名のデータをテキストエリアに呼び出しできます。

【まとめ】

次の手順のように、BASICプログラムで書き込んだラムデータファイルのデータをパソコン等へ送出できます。なお、パソコンと接続するためには、ケーブルが必要です。(371ページ付録参照)

① TEXT R I でファイルを確保
② BASICでデータを登録（書き込み）
③ TEXT R L でテキストエリアに呼び出し
④ TEXT E で内容の確認

⑤ TEXT S F でフォーマット設定
⑥ TEXT S S でパソコン等へ出力

(注) ● BASICのPRINT＃命令でデータを書き込んだときのデータのフォーマットについては、PRINT＃命令を参照してください。CR、LFコードはテキストプログラムでの行の区切りになります。
なお、データの最初が数字の文字データ（例"123"）の場合は、TEXTモードでは行番号との区別がつかず、正しく編集できません。

ご注意

- 変換元のプログラムを消去しながら変換していっても、なおメモリが足りなくなるとエラー（MEMORY　OVER）になります。
この場合、変換された部分と、変換できなかった部分がTEXTとBASICに分かれてしまいます。したがって、変換する前にプログラムをプリンタに印字しておくことをお勧めします。
特にBASICからTEXTに変換するときに、起こる可能性が高いので注意してください。
- パスワード（PASS命令参照）が設定されているときは、BASICコンバータに入れません。あらかじめ、RUNまたはPROモードでパスワードを解除してください。
- BASICコンバータでは、データエリア（変数）を消去しません。フリーエリア（空きエリア）の容量が少ないと、変換できなくなることがあります。あらかじめCLEAR命令で消去するなど、フリーエリアを広げておいてください。
- TEXTからBASICに変換するときは、TEXTの内容が何であっても、それを本機用のBASICプログラムと見なして変換作業を行います。したがって、TEXTの内容によってはBASICに変換して、再度TEXTに変換しても、元の内容に戻らない場合があります。

〈例〉 TEXT　10FORMULA
　　↓（変換）
BASIC　10：FOR␣MULA
　　↓（変換）
TEXT　10FOR␣MULA

第6章
C言語機能

C言語はＵＮＩＸ※というオペレーティングシステムを記述するために生まれました。しかし、今ではパソコンはもちろん、ワークステーションやミニコンでのポピュラーなプログラミング言語として定着しています。

C言語は、基本的な部分の統一性が高いため、異機種間でのプログラミングの移植性がたいへん高く、本機で開発したプログラムであっても、少しの手直しでパソコンなどで実行できるという特徴があります。

本章では、C言語で書かれたプログラムを本機で実行する方法および一般的なC言語との違いについて説明します。

C言語自身については、入門書等の関連書籍が豊富に市販されていますので、それらの書籍を参照してください。

※ＵＮＩＸオペレーティングシステムはUNIX System Laboratories, Inc. が開発しライセンスしています。

1．C言語の特徴

C言語の特徴は、言語自体がコンパクトでありながら、ハードウェアの制御やシステムプログラムからアプリケーションプログラムまで扱える守備範囲の広さにあります。

■ハードウェアに密着したプログラムが作れる

ＢＡＳＩＣやＦＯＲＴＲＡＮのような高級言語でありながら、ビット演算や、メモリのアドレスが直接扱えます。これらの機能により、これまでアセンブラ言語に頼ってきたハードウェアの操作や外部機器の制御が、C言語によって記述できるようになりました。そのため、これらのプログラムの開発が効率的に行えるようになりました。

■高級言語としての機能を備えている

構造化プログラミングによる、読みやすく、わかりやすいプログラムの記述ができます。そのため、プログラムの開発が効率的に行えます。
また、数値処理やデータ処理のためのデータ型が豊富に用意されています。そのため、科学計算プログラムやアプリケーションプログラムなど、幅広い目的のプログラミングが可能です。

■コンパクトな記述ができる

プログラムをコンパクトに記述するための機能（演算子）が豊富にそろっています。また、ポインタ操作によって実行効率の高いプログラムが作れます。

■移植性が高い

ハードウェアに密着したプログラムの移植性が高いことがC言語の流行の大きな要因です。異機種間でのCプログラムの移植性も高く、本機で開発したプログラムであっても、少しの手直しでパソコンなどで実行できるようになります。

このように大変強力なC言語ではありますが、少々欠点もあります。
それは、機能が豊富なため難解なプログラムができたり、システム自体が暴走するプログラムが記述できるという危険性があることです。

ご注意
したがって、機械語プログラムと同様、消えては困るプログラムやデータは、紙に書き写しておいてください。CE-126P（プリンタ）をお持ちの場合は、印字しておいてください。

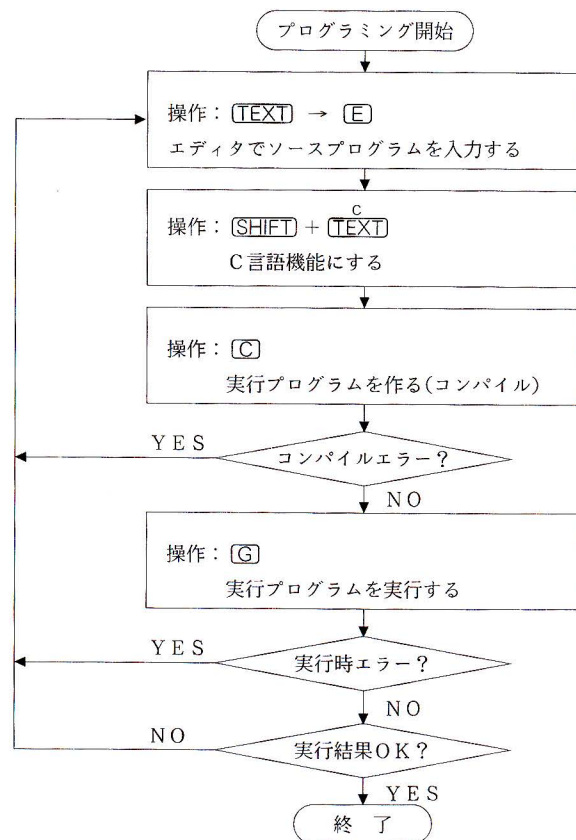
2．始めようＣプログラミング

ここでは本機でのプログラミングの方法と、Ｃプログラムの基本的な約束ごとを説明します。

2．1　Ｃプログラムの作成から実行まで

Ｃ言語などのプログラムは人間が理解しやすい言葉、約束ごと（文法）で記述されます。しかし、Ｃ言語のままではコンピュータは直接理解できません。

そのため、Ｃ言語で記述されたプログラムは、コンピュータが理解できる機械語に変換する必要があります。

したがって、本機ではソースプログラムの作成（Ｃ言語でのプログラミング）、コンパイル（変換）、実行という手順でプログラミングします。

■ステップ１：ソースプログラムを作る

メッセージを出力する簡単なプログラムを作ってみましょう。最初にＣ言語でソースプログラムを作ります。

◆テキストエディタを呼び出す

ＴＥＸＴ（テキスト）モードに入り、エディタ機能を呼び出します。

TEXT → E

```
TEXT EDITOR
<
```

◆ソースプログラムを入力する

次のプログラムを入力します。

```
10␣main ()↵
20␣{↵
30␣␣printf ("C␣プログラミング¥n");↵
40␣}↵
```

CAPS を押して小文字入力モードにします（“ＣＡＰＳ”を消灯させます）。

行の先頭に行番号を入力してから各行のプログラムをキー入力してください。カナ入力は、カナ を押してからローマ字で入力します。英字入力に戻るにはもう一度 カナ を押します。

各行の終わりでは必ず ↵ を押してください。行番号30は画面上では１行に収まらず２行に表示されますが、↵ が押されるまでは１行として扱われます。␣は SPACE の入力を意味します。

もし、入力をまちがえた場合などは、「第５章　ＴＥＸＴモード」を参照して修正してください。

一般のプログラムでは行番号をつけるとエラーになります。しかし、本機のテキストエディタは行番号をつけて入力しなければならないため、このような記述となっています。ほかのＣ言語処理系では、行番号は取り除かなければなりません。

> **ご注意**
> この項（2．始めようＣプログラミング）では、プログラム中にスペースマーク ␣ や ↵ を入れています。

■ステップ２：ソースプログラムをコンパイルする

ソースプログラムをコンパイルし実行プログラムを作ります。

◆Ｃ言語機能のメニュー画面にする

SHIFT を押しながら TEXT を押すと、Ｃ言語機能のメニュー画面が呼び出されます。

SHIFT ＋ TEXT

```
*** C ***

Compile Trace Go Stdout
```

各コマンドは次のような働きをします。

- Compile：ＴＥＸＴで作成したプログラムをコンパイルする
- Trace　：コンパイルしたプログラムをトレース実行する
- Go　　 ：コンパイルしたプログラムをノーマル実行する
- Stdout ：出力を画面 ↔ プリンタとで切り替える
  （プリンタ動作可能時、S を押すたびに切り替わる）

各コマンドはコマンドメニューの先頭文字を入力することで実行されます。

◆コンパイルする

[C] を押してください。コンパイルが実行されます。

[C]

コンパイル実行中は compiling と一時表示し、次の画面に移ります。ただし、プログラムが短いときは、この表示は一瞬で終わります。

◆コンパイルが終了すると

コンパイルが終了すると次の画面が表示されます。

```
*** C ***
Compile Trace Go Stdout
complete !
```

プログラムに悪いところがあると、エラーのある行番号とエラーの内容が表示され、コンパイルが中断されます。このような場合は [TEXT] [E] と押してテキストエディタに戻り、プログラムを修正してからもう一度コンパイルしてください。

（注）memory full と表示されたときは、コンパイルに必要なメモリが足りません。不要になったＢＡＳＩＣの変数やプログラムを消去してから、もう一度コンパイルしてください。

■ステップ３：プログラムを実行する

プログラムの実行にはノーマル実行と、トレース実行の２つのモード（機能）があります。ここではノーマル実行を行います。またどちらのモードも実行プログラムが作成されていないときは、自動的にコンパイルをして実行プログラムを作り、実行します。

トレース実行については「２．２　プログラムのトレース実行」で説明します。

◆ノーマル実行する

コンパイルが終了したら [G] を押してください。プログラムが実行され、画面には次のように表示されます。

[G]

```
C ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ
*EXIT (40)
```

＊ＥＸＩＴ　（４０）はプログラムの実行が終了した行番号を示しています。[CLS]、[↵]、[BREAK] でメニューに戻ります。

◆実行時エラーが発生したら

実行時にエラーが見つかることがあります。実行時エラーが発生すると、エラーメッセージを出力して実行を停止します。コンパイルエラーと同様に、エディタを呼び出してソースプログラムを修正してください。

実行時エラーメッセージについては243ページを参照してください。

２．２　プログラムのトレース実行

プログラムの構造が複雑になると、エラーが発生するときや、期待どおりの結果が得られないときの処理が大変になります。エラーを発見するために便利なのがトレース実行です。

トレース機能とはプログラムを１ステップずつ実行していく機能のことです。

各ステップごとに変数の値を調べ、エラーの原因を探していきます。

ここでは、次のプログラムを例に、トレース実行します。テキストエディタで入力して、Ｃ言語機能でコンパイルしてください。

```
10 main() [↵]
20  { [↵]
30   int i, gokei = 0 ; [↵]
40   for (i = 1 ; i < 51 ; i++)  { [↵]
50     gokei += i ; [↵]
60     printf ("1+...+%d = %d¥n", i, gokei) ; [↵]
70   } [↵]
80  } [↵]
```

■トレースモード

Ｃ言語機能のメニュー画面で [T] を押し、トレースモードにします。

画面には次のように、最初に実行する行が表示されます。

```
?10 main()
```

[↵] を押せば、？の行を実行し、次のステップへ進みます。順次 [↵] で実行を進めます。

[BREAK] を押せば、ブレークモードに入ります。

■ブレークモード

ブレークモードでは次のキー操作ができます。

[↵]：実行を再開します。

[C]：実行を再開します。

[A]：実行を中断してメニュー画面に戻ります。

[T]：トレースを指定して、実行を再開します。

[N]：トレースを解除して、実行を再開します。

[D]：変数表示モードに入ります。

変数表示モードでは変数名を入力すると、そのステップでの変数の内容が表示されます。変数ｉの値を表示させてみましょう。

[BREAK]

ブレークモードになります。

[D] i [↵]

```
1+...+1 = 1
 40  for(i=1;i<51;i++)  {
Break>D
ﾍﾝｽｳ>i
int : 2 (0x0002)
ﾍﾝｽｳ>_
```

ｉの値が２の場合

［補足］・T を押してトレースモードにした後、↵ を5回押すと i はカウントアップされ2になります。

・i は必ず小文字（“CAPS”消灯）で入力してください。プログラムでの指定が小文字のためです。D を押すときは“CAPS”が消灯していても有効です。

画面には i のデータ型（int）とその値が表示されます（カッコ内は16進表示で前に 0x がつきます）。
変数表示モードは BREAK を押せば終了し、ブレークモードに戻ります。

2．3　表示出力と印字出力の切り替え

CE-126P（プリンタ）をお持ちの場合は、本機とプリンタが接続され、動作可能状態になっているとき、C言語機能のメニュー画面で S を押すと、画面の **Stdout** が **Stdprn** に切り替わります。
もう一度 S を押すと **Stdout** に戻ります。
この切り替えは、下記の出力関数で、出力 stream が stdout に指定されている場合に、出力先を画面とプリンタとで切り替えるときに使用します。
出力 stream が stdprn に指定されているときは、メニュー画面での指定にかかわりなく、プリンタに出力されます。

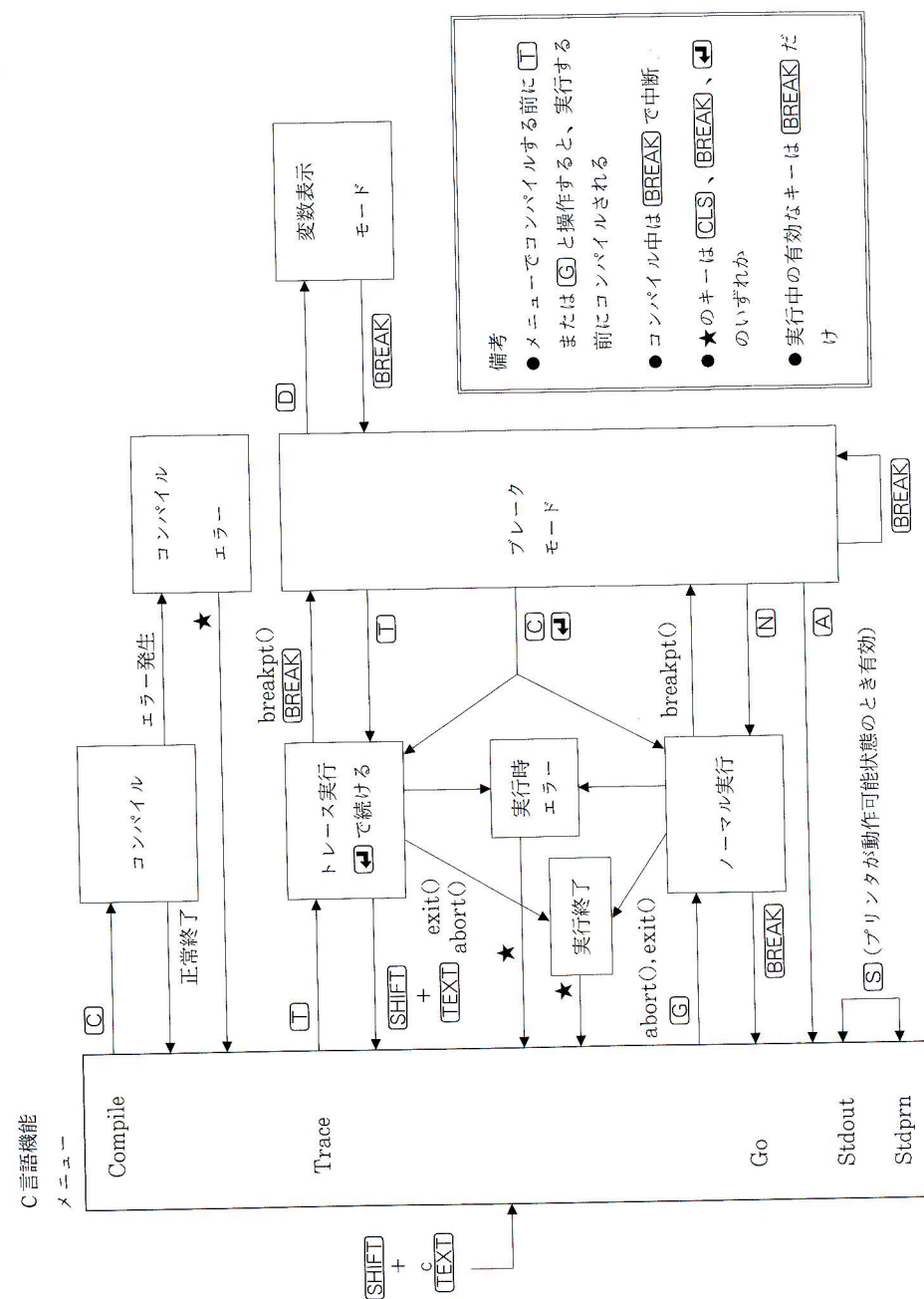
Stdout 表示時：画面に出力
Stdprn 表示時：プリンタに出力
C言語機能選択時などには Stdout になります。

〈有効な関数〉

putc
fputc
fputs
fprintf

ご注意

Cプログラム実行中は、電池が消耗しても“BATT”が点灯しませんので、実行させたまま長時間放置しないでください。実行させたまま長時間放置しますと、電圧が低下し、正常な動作をしなくなる恐れがあります。
なお、実行中に誤動作等が発生した場合は電池の消耗が考えられます。BERAK またはリセットスイッチを押して実行を止め、“BATT”の点灯または電源が切れる場合は、速やかに電池を交換してください。

2．4　C言語機能一覧

２．５　Ｃプログラムのスタイル

ＢＡＳＩＣになれている人は、Ｃ言語のプログラムはとても雰囲気が違うと感じているかもしれません。その理由のひとつは、Ｃ言語のプログラムは小文字を中心に記述されることにあると思います。Ｃプログラムでは特別な場合を除いて大文字は使いません。
そのほかにもＣ言語特有のプログラミングスタイルがあります。ここではそのうちの基本的な約束ごとを説明します。
テキストエディタを呼び出して、次のプログラムを入力してください。

```
１０␣main ()↵
２０␣ {↵
３０␣␣printf ("タノシイ␣") ;↵
４０␣␣printf ("Ｃ␣プ゜ログ゛ラミング゛￥n") ;↵
５０␣}↵
```

■基本的なスタイル

Ｃプログラムの最も基本的な構成は次のようになります。

```
main ()
 {
  実行文
 }
```

main は文字どおり、Ｃプログラムのメイン（主プログラム）であることを示しており、プログラムの実行は、必ず main () から始まります。main () で実行する内容は ｛と ｝ との間に書きます。
これがＣ言語の基本的な約束で、プログラムの実行の始まりと終わりがはっきりとわかるような形式になっています。
入力したプログラムで実行されるのは、次の２つの出力文です。

```
printf ("タノシイ␣") ;
printf ("Ｃ␣プ゜ログ゛ラミング゛￥n") ;
```

printf 文は " " で囲まれた文字の並び（文字列）を出力する命令（Ｃ言語では関数という）です。

■実行の区切り

それぞれの実行文の終わりに ；（セミコロン）がついていることに注目してください。；は一つの実行単位（文）の終わりを示す大切な目印です。初心者は ；を忘れがちです。気をつけてください。
；を忘れると、次の行以降でエラーになります。

ＢＡＳＩＣでは行が実行の単位（区切り）を意味していましたが、Ｃプログラムは行という単位にしばられないで記述できます。これをフリーフォーマットといいます。
先のプログラムは次のように書くことができます。

```
main () {printf ("タノシイ␣") ; printf ("Ｃ␣プ゜ログ゛ラミング゛￥n") ;}
```


同じ働きをするプログラムでも、書きかたによって読みやすさ、わかりやすさが違ってきます。フリーフォーマット方式では、読みやすいプログラムを作ることを、常に心がけておくことが大切です。

■出力画面の改行

プログラムをコンパイルし、実行してください。次のように出力されます。

```
タノシイ Ｃ プ゜ログ゛ラミング゛
*ＥＸＩＴ (５０)
```

実行結果を見ると ￥n が画面に表示されていません。￥n は改行を意味する特別な文字で、Ｃプログラムでは文字列中の￥n のある位置で改行が実行されます。ＢＡＳＩＣのＰＲＩＮＴ文と違って、自動的な改行は行いません。
最初の printf 文を次のように変更してみましょう。

```
printf ("タノシイ￥n") ;
```

画面は次のように２行に出力されます。

```
タノシイ
Ｃ プ゜ログ゛ラミング゛
*ＥＸＩＴ (５０)
```

参　考

ソースプログラムのセーブ（保存）

入力したＣプログラム（ソースプログラム）は、本体のプログラムファイルエリアへ登録して保存します。
[TEXT] を押してテキストの機能選択画面にした後、[F] を押すとプログラムファイルのメニュー画面になります。
この画面で [S] を押すとセーブ（Save）が選ばれ、ファイル名の入力待ちになります。
ファイル名を入力して [↵] を押すと、プログラムがファイルされます。
なお、Ｃ言語のプログラムは、一般的にファイル名の拡張子を ．Ｃ とします。
　〈例〉 ＦＩＬＥ ＮＡＭＥ＝ＴＥＳＴ．Ｃ
このファイルをテキストエディタへ読み込むときは、Load を使用します。
くわしくは、「第５章　ＴＥＸＴモード」を参照してください。

３．基本的なＣプログラミング

この章では数値計算のプログラムを例に、Ｃプログラミングにおける変数の扱いかたを説明します。数値を出力する方法、変数宣言といった、Ｃプログラミングの基本的な機能を説明します。

３．１　整数を扱うプログラム

■数値の出力

辺の長さが10の正方形の面積を求めるプログラム例です。

```
１０　main ()
２０　 {
３０　　printf ("ヘン　１０　ノ　メンセキハ　%d　デ゛ス¥n", １０＊１０)；
４０　}
```

プログラムを実行すると、画面には次のように出力されます。

```
ヘン　１０　ノ　メンセキハ　１００　デ゛ス
＊EXIT　(４０)
```

printf 文の文字列の %d の部分が、１００に変わって出力されています。この１００という数値は10×10の演算結果です。%d は printf 文の中の文字列の後ろに続く値に置き換えられます。

```
printf ("ヘン　１０　ノ　メンセキハ　%d　デ゛ス¥n", １０＊１０)；
```

printf 文は出力する文字列の中に%記号があると、文字列と ，で区切られたデータをその位置に出力します。%記号に続く文字ｄは、データの表示形式を指示するための変換文字で、「データを10進整数で表示する」ことを意味します。

このように printf 文の文字列は、単なる文字列と違い画面への表示形式を示す働きをするもので、書式文字列と呼ばれます。

%d のような働きをするものを変換指定といいます。整数のための変換指定には次の種類があります。

変換指定	表示形式
%d	１０進数に変換する
%x	１６進数に変換する
%o	８進数に変換する

■変数を使う

変数を使って、上記のプログラムをもう少し一般的なプログラムにします。

```
１０　main ()
２０　 {
３０　　int　hen, menseki；
４０　　hen＝１０；
５０　　menseki ＝hen＊hen；
```



```
６０　　printf ("ヘン　%d　ノ　メンセキハ　%d　デ゛ス¥n", hen, menseki)；
７０　}
```

変数はいろいろなデータを記憶しておくための箱のようなものです。この箱には名前と、記憶しておくデータの種類（型）が決められます。これを変数宣言といいます。

行番号３０で、変数の宣言をしています。ＢＡＳＩＣと違って、Ｃ言語ではプログラムで使う変数は、あらかじめ名前と変数の種類（データ型）を明確に宣言してからでないと使えません。変数宣言は次のように、データ型の指示に続いて変数名を書きます。

```
int　hen,　menseki；
 ↑    ↑       ↑
 │    └───────┴──── 変数名
 │
データ型
```

int は、変数 hen および menseki の扱うデータが整数値（小数点のつかない数値）であることを示すキーワードです。同じ型の変数を複数宣言する場合は各変数名を ，で区切ります。宣言の終わりには ；が必要です。

行番号４０は代入文です。＝ は代入を意味する演算子で、右辺の値が左辺の変数に格納されます。

```
[ １０ ]　←　１０
  hen
```

行番号６０の printf 文で変数 hen と menseki の内容を出力します。

```
ヘン　１０　ノ　メンセキハ　１００　デ゛ス
＊EXIT　(７０)
```

printf 文では、変換指定の位置に変数の値が表示されます。複数の変数を出力する場合は、変数を ，で区切ります。printf 文は記述された順に変数の値を表示します。

```
printf ("ヘン　%d　ノ　メンセキハ　%d　デ゛ス¥n", hen, menseki)；
```

■データの型と数値の範囲

Ｃ言語では小数点のつかない数値を整数型、小数点を含む数値を実数型（浮動小数点型ともいう）として、明確に区別します。これはデータ型によってコンピュータ内部で表現される形式が違うためです。また整数型、実数型はそれぞれ表現する数値の大きさによって、さらに分類されます。代表的な型を次に紹介します。（数値の範囲はコンピュータによって異なります。）

分類	データ型	扱う数値の範囲	ビット長
整数型	char	−128〜＋127	8ビット
	int	−32768〜＋32767	16ビット
	short	−32768〜＋32767	16ビット
	long	−2147483648〜＋2147483647	32ビット
実数型	float	$\pm 1 \times 10^{-99}$ 〜 $\pm 9.999 \times 10^{+99}$	32ビット
	double	float の仮数部有効桁が10桁	64ビット

整数型の前に次の修飾子をつけることができます。

unsigned　正負の符号なし、つまり正の数のみ扱う

double 型に修飾子 long をつけることができますが、本機では double と同じ扱いになります。くしくは「６．Ｃ言語の要点」を参照してください。

（注）変数を使用する場合、各型で扱える数値の範囲を超えるような使いかたをすると正しい結果が得れません。

■変数の名前

変数の名前のつけかたに制限があります。

- 英文字で始まり、英字または数字で表す（カナ文字は使えない）
- 大文字と小文字は区別される
- 下線（＿）は英字として扱う
- ＋、－、＆など、特殊記号は使えない
- 名前の長さは31文字まで有効（32文字以降は無視）
- int、double といった、Ｃ言語のキーワードは使えない

正しい例

__max	：＿は英字に含む
int__var	：キーワードは名前の一部に使える
MyName	：大文字と小文字が混在してもよい
number 2	：先頭でなければ数字が使える

エラーになる例

float	：キーワードは使えない
int－var	：－は特殊記号である
ヘンスウａ	：カタカナは使えない
１０ｎ	：先頭に数字は使えない

本機で用意されているライブラリ関数名も名前に使えません。キーワード、ライブラリ関数については「６．Ｃ言語の要点」、「７．ライブラリ関数」を参照してください。

Ｃプログラミングでは、変数名は基本的に小文字で表します。また変数の名前はその働きがわかる名前をつけることが、わかりやすいプログラミングのコツです。

■キーボードからの数値入力

辺の長さをキーボードから入力するプログラムを示します。

```
１０  main ()
２０   {
３０    int  hen, menseki;
４０    printf ("ヘンノ　ナガサ?") ;
５０    scanf ("%d", &hen) ;
６０    menseki=hen*hen;
７０    printf ("ヘン　%d　ノ　メンセキハ　%d　デス¥n", hen, menseki) ;
８０  }
```


行番号５０の scanf 文が入力文で、キーボードからの入力を実行します。

scanf（"%d", &hen）；

この文が実行されると、プログラムはキーボードからの入力待ちの状態になります。キーボードから数値を入力すると、そのデータは変数 hen に格納されます。

%d は printf 文の場合と同様に、入力されたデータを10進整数として扱うことを指示する書式文字列です。

数値が格納される変数の名前に＆記号がついていることに注目してください。

&hen

& はアドレス演算子と呼ばれるもので、**&hen** は変数 hen の記憶されている場所（メモリの番地）を表します。scanf 文では入力されたデータを格納する「場所」を指示しなければなりません。

scanf 文はＢＡＳＩＣのＩＮＰＵＴ文と違って、書式文字列の中に入力を促すメッセージを記述することはできません。そのため、行番号４０のように printf 文でメッセージを出力する必要があります。

３．２　実数を扱うプログラム

■実数を使う

３教科のテストの平均点を求めるプログラムで、実数の使いかたを説明します。

```
１０  main ()
２０   {
３０    int  k=７６, /*  コクゴ  */
４０         s=８８, /*  スウガク */
５０         e=８１; /*  エイゴ  */
６０    double  ave;
７０    ave= (k+s+e) /３．０;
８０    printf ("ヘイキンテン:%f¥n", ave) ;
９０  }
```

行番号３０、４０、５０が各教科の点数を入れるための変数宣言です。ここでは変数の宣言と同時に代入を行っています。これを変数の初期化といいます。またそれぞれの行に ／＊ と ＊／ で囲まれたメッセージが記述されています。これをコメントといいます。コメントはプログラムの実行には何の影響も与えません。コメントはプログラムをわかりやすく記述するための働きをします。

行番号７０で平均点を求めています。３教科の平均点ですから、合計を３で割ればよいのですが、わざわざ３．０と小数点をつけて割っています。これは３で割った場合は整数どうしの演算ということになり、結果も整数、つまり小数点以下が切り捨てられてしまいます。これを防ぐために３．０という実数型で割ることで、演算が実数型の演算となり、結果も小数点以下の値が求められます。

printf 文の変換指定に **%f** が使われています。%f（および%e、%g）は double 型のデータを出力するための変換指定です。%f は次のように表示形式を指定できます。

%６．２f

これは正負の符号、小数点を含めた全体の桁数を６、小数点以下を２桁に指定することになります。実数型には%f 以外にも **%e**（指数形式）、**%g** の変換指定があります。くわしくは「７．ライブラリ関数」の printf 関数を参照してください。

■指数を使う

科学計算、たとえば地球と太陽との距離のように大きな数値や、原子の重さのように非常に小さい数値を扱うには指数を使った計算が必要になります。

たとえば太陽からの光が地球に届く時間を求めてみると太陽と地球との距離、光の速度は次のように表します。

太陽と地球との距離　1.496×10^{8}〔km〕　→１．４９６e８

光の速度　2.998×10^{5}〔km／sec〕→２．９９８e５

```
10  main ()
20  {
30    double d=1.496e8;
40    double c=2.998e5;
50    double t=d/c;
60    printf ("time=%7.2f [sec] ¥n", t);
70  }
```

```
time= 499.00[sec]
*EXIT (70)
```

出力の変換指定を％７．２ｆとしたため、出力した数値の前が１桁空白になっています。桁数を指定すると、数値は右詰めで表示されます。

４．プログラムの流れを制御する

プログラムは上から順次実行されるとは限りません。プログラム中で与えられた条件によって実行される文が選択されたり、繰り返しが行われたりします。このようなプログラムの流れを制御する構文を説明します。

４．１　条件文の使いかた

条件文は与えられた条件によって実行の流れを枝別れするための構文です。

■ if 文

if 文は次のようになります。

```
if (条件式)  /*この条件が成立すれば*/
  文1;      /*文1を実行し       */
else        /*それ以外ならば    */
  文2;      /*文2を実行する     */
```

実行される文は１つの文（単文という）でも、複数の文を含んでもかまいません。複数の文の場合は、それらの文を中カッコ、つまり ｛と｝ で囲みます。これを複文、またはブロックといいます。

else 以下を省略できます。この場合、条件が不成立なら何もしないで条件文を抜けることになります。

分数では分母が０の場合は演算不能で、実行時エラーが発生します。次はこれを避けるために if 文を使ったプログラムです。

```
10  main ()
20  {
30    double a, b;
40    printf ("a/b ヲ モトメル¥n");
50    printf ("a? ");
60    scanf ("%lf", &a);
70    printf ("¥nb? ");
80    scanf ("%lf", &b);
90    if (b==0)
100       printf ("¥n0 デ ワレマセン¥n");
110   else
120       printf ("¥na / b = %f¥n", a/b);
130 }
```

キーボードから変数ａ、ｂに数値を入力してください。行番号９０の条件式ｂ＝＝０で、ｂの値が０であるかどうかを調べます。ｂの値が０以外なら割り算を実行し、０ならばエラーメッセージを出力してプログラムは終了します。

等しいかどうかをチェックする演算子は、等号 ＝ を２つ続けます。C言語では「等しい」ことを意味する ＝＝ と、「代入」を意味する ＝ とは明確に区別します。

実行される文はキーワード if または else より行頭を数桁下げます（本書では２桁）。これは実行のレベルが違うことを明らかにすることで、プログラムをわかりやすくするためのテクニックです。これを段付けといいます。

■大小を比較する演算子

条件を与える式に、基本的には関係演算子を使います。関係演算子は２つの値を比較して、等しいか等しくないか、数値的にどちらが大きいか小さいかを調べる演算子です。

式	真になる場合
a＝＝b	aはbに等しい
a！＝b	aはbに等しくない
a＜b	aはbより小さい
a＞b	aはbより大きい
a＜＝b	aはb以下である
a＞＝b	aはb以上である

関係演算子は演算の結果として、真には数値１、偽には０を返します。

■ switch ～ case 文

条件による分岐先が多くなる場合は switch ～ case 文を使うとわかりやすくなります。switch ～ case 文は次のようになります。

```
switch （整数式）      ｛
    case  定数１：     ／＊ 整数式の値が定数１なら ＊／
      文１；break；    ／＊ 文１を実行する ＊／
    case  定数２：     ／＊ 整数式の値が定数２なら ＊／
      文２；break；    ／＊ 文２を実行する ＊／
          ～                   ～
    case  定数ｎ：     ／＊ 整数式の値が定数ｎなら ＊／
      文ｎ；break；    ／＊ 文ｎを実行 ＊／
    default：          ／＊ それ以外なら ＊／
      文；             ／＊ 文を実行する ＊／
｝
```

break は無条件で制御構文を抜けるためのキーワードです。

次は暦の月を英語で表示するプログラムです。ただしスペースの関係で３月までの表示とします。

```
１０    main ()
２０     ｛
３０      int  month；
４０      printf（"ナンガツ  ？  "）；
５０      scanf（"％ｄ"，＆month）；
６０      switch  （month）  ｛
７０          case  １：
８０              printf（"January￥ｎ"）；break；
９０          case  ２：
１００            printf（"February￥ｎ"）；break；
１１０        case  ３：
１２０            printf（"March￥ｎ"）；break；
１３０        default：
１４０            printf（"ソノタ￥ｎ"）；
１５０      ｝
１６０  ｝
```

４．２ 繰り返し文の使いかた

同じ作業を繰り返し実行するには、繰り返し文を使います。Ｃ言語の繰り返し文には for 文、while 文、do ～ while 文が用意されています。

■ while 文

while 文は与えられた条件式が成り立つ間、文を実行します。


```
while（条件式）
    文；
```

if 文と同様、実行される文は単文でも、複文でもかまいません。

キーボードから数値を繰り返し入力し、その合計を求めるプログラムです。入力するデータの個数もキーボードから入力します。

```
１０    main ()
２０     ｛
３０      int  data，num；
４０      int  i＝０，sum＝０；
５０      printf（"データノ  コスウ？  "）；
６０      scanf（"％ｄ"，＆num）；
７０      while（i＜num）  ｛                    ← 複文の始め
８０        printf（"￥ｎデータ？  "）；
９０        scanf（"％ｄ"，＆data）；
１００      sum＝sum＋data；
１１０      i＝i＋１；
１２０    ｝                                     ← 複文の終わり
１３０    printf（"￥ｎゴウケイ  ％ｄ￥ｎ"，sum）；
１４０  ｝
```

行番号１００は「sum の内容と data の内容をたして、その結果で sum の内容を書き換える」ことを意味します。つまり入力データを順次、sum に合計していきます。行番号１１０も同様で、この文が実行されるたびに、変数ｉの内容が１ずつ増加していきます。
行番号７０の条件式では、ｉと num とを比較しｉの値が num と等しくなった時点で繰り返しを終了し、行番号１３０に実行が移ります。

■代入演算子とインクリメント演算子

Ｃ言語では行番号１００の文を次のように記述できます。

```
sum  ＋＝  data；
```

この記述法は、記述の簡素化だけでなく、プログラムの実行速度が速くなることが期待できます。このような演算子を代入演算子といいます。代入演算子は＋演算子だけでなく－、／、＊、％（割り算の余りを求める）演算子など、他の演算子にも適用できます。
また行番号１１０は次のように記述できます。

```
i＋＋；
```

これをインクリメント演算子といい、この文が実行されるたびにｉの内容は１ずつ増加します。インクリメント演算子はＣプログラミングでは多用されます。
代入演算子、インクリメント演算子については「６．Ｃ言語の要点」を参照してください。
（注）代入演算子を使用する場合で、右辺と左辺でデータ型が異なるときは、右辺の結果を左辺の型に変換した後で演算をするため、一般のＣ言語と結果が異なる場合があります。型の違うデータの演算は算術演算子による記述を行ってください。

■ for 文

for 文の一般的な書式は次のようになります。

```
for（文１；条件式；文２）
    本文；
```

while 文を次のように記述した場合と全く同じ働きをします。本文は単文でも、複文でもかまいません。

```
文１；                ← 繰り返しに入る前に実行される
while（条件式）  {    ← 条件が成立している間、｛｝を繰り返し実行する
    本文；
    文２；            ← 繰り返しを行うたびに実行される
}
```

文１で繰り返しを制御する変数（制御変数という）を宣言し、初期化します。条件式で制御変数のチェック、文２で制御変数のカウンタの働きをさせます。

```
１０  main ()
２０   {
３０    int  i；
４０    for（i＝０；i＜１０；i＋＋）  {
５０        printf（"i＝％d␣␣␣"，i）；
６０    }
７０  }
```

プログラムを実行すると、変数ｉの値が０から９まで変化していくことが確認できます。行番号４０で繰り返しの制御が行われます。行番号４０は「変数ｉの初期値を０とし、ｉが１０より小さい間、ｉを１ずつ増加しながら繰り返す」という意味になります。

■制御文の中の制御文

条件文、繰り返し文はそれぞれ独立した１つの文ですから、条件文や繰り返し文の中で実行される文として使うことができます。これをネスティングといいます。

次はかけ算の九九を出力するプログラムです。for 文の中で for 文が実行されています。

```
１０    main ()
２０     {
３０      int  i, j；
４０      for（i＝１；i＜１０；i＋＋）  {
５０          for（j＝１；j＜１０；j＋＋）  {
６０              printf（"％d  ＊  ％d  ＝  ％d  ", i, j, i＊j）；
７０              printf（"￥n"）；
８０          }
９０      }
１００  }
```


参　考

上記のプログラムを実行すると、スピードが速すぎて表示の確認ができません。
このようなときは、for 文を使って表示時間を調整する方法があります。
たとえば、前記のプログラムに、次の行を追加します。

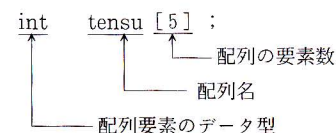
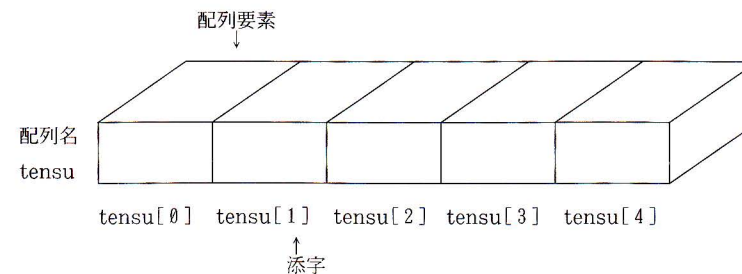
```
３５    int  k；
６５        for（k＝０；k＜１０００；k＋＋）；
```

65行は、「ｋが０から１０００になるまで繰り返す」という意味です。
つまり、65行を繰り返し実行している間は、画面が変わりません。
１０００を他の値に変えれば、表示時間を変えることができます。

５．Ｃ言語らしいプログラミング

次に配列、関数などのＣプログラミングでの特徴的な機能を中心に説明します。

５．１　配列

配列は同じ種類（データ型）のデータを大量に処理するとき、威力を発揮します。Ｃプログラミングでは変数宣言と同様に、配列の宣言が必要です。

これで配列名が tensu 、配列要素のデータ型が int 、配列要素の数が５である配列が宣言されたことになります。実際には配列のための領域がメモリに確保されます。

配列を宣言してから、その配列の各要素を指定するときは、配列名の後ろに大カッコ［ ］で囲んだ数値を書いて指定します。この数値を添字といいます。それぞれの配列要素は単独の変数と同じように扱えます。たとえば、配列 tensu の３番目の要素に代入するには次のようにします。

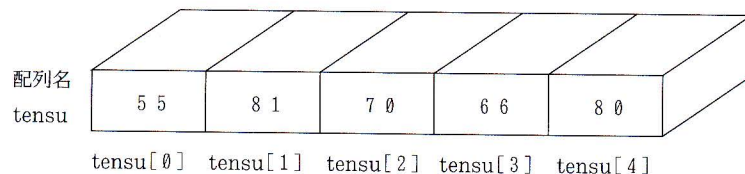
```
tensu［２］＝７０；
```

３番目の要素でありながら、添字は２になっていますがまちがいではありません。C言語では配列の要素は１ではなく０番目から始まるために、このような表記になるのです。

■配列の操作

次は、あるテストの得点での最高点を求めるプログラムです。

```
１０　main ()
２０　{
３０　　int　tensu［５］；
４０　　int　i, n, max＝０；
５０　　tensu［０］＝５５；
６０　　tensu［１］＝８１；
７０　　tensu［２］＝７０；
８０　　tensu［３］＝６６；
９０　　tensu［４］＝８０；
１００　for　(i＝０；i＜５；i＋＋)　{
１１０　　if　(max＜tensu［i］)　{
１２０　　　max＝tensu［i］；
１３０　　　n＝i；
１４０　　}
１５０　}
１６０　printf ("サイコウハ　%dバン：%dテン¥n", n, max)；
１７０　}
```

各点数は次の図のように各要素に格納されます。

プログラムでは、行番号１００から１５０において、配列要素を０番目から順にそれ以前の最大値maxと比較し、より大きい値でmaxの内容を書き換えていきます。for文を抜けた後のmaxに、配列要素の中の最大値が残されます。

行番号３０および５０から９０までの代入文は次のように記述できます。

```
int　tensu［５］＝{５５，８１，７０，６６，８０}；
```

このように配列宣言のときに各要素に一括して代入できます。これを配列の初期化といいます。

ただし、一般のC言語では、このように書けないものがあり、このように書くとエラーになる場合があります。

■文字の配列

Cプログラミングでは文字列は文字の配列として扱います。

```
１０　main ()
２０　{
３０　　char　str［７］＝"string"；
４０　　int　i；
５０　　printf ("%s¥n", str)；
６０　　for (i＝０；i＜６；i＋＋)
７０　　　printf ("%c␣", str［i］)；
８０　}
```

行番号３０で、文字の配列strを宣言すると同時に、文字列の定数 string で初期化しています。Cプログラミングでは文字と文字の並びである文字列は区別しなければなりません。文字定数の場合はシングルクォーテーションマーク' 'で１文字を囲みます。文字列定数はダブルクォーテーションマーク" "で文字の並びを囲みます。

```
char　c＝'A'；　　　　　　：文字定数
char　str［７］＝"string"；　：文字列定数
```

行番号５０で文字列strの内容を出力します。%s が文字列のための変換指定です。

行番号６０、７０で文字列の要素を順次出力します。%c が１文字を出力するための変換指定です。なおfor文は繰り返し実行する文が単文のため { } は省略しています。

■文字コード

文字の配列がどのようになっているか、もう少しくわしく調べましょう。

```
１０　main ()
２０　{
３０　　char　str［３］＝"AB"；
４０　　int　i；
５０　　for (i＝０；i＜３；i＋＋)
６０　　　printf ("%c：%x¥n", str［i］, str［i］)；
７０　}
```

画面には次のように表示されます。

```
A：41
B：42
：0
*EXIT　(70)
```

文字にはそれぞれ文字コードが決められており、文字配列の各要素には文字コードを意味する数値が格納されます。画面の１行目、２行目は文字配列の要素'A'、'B'とその文字コードが16進数で表示されます。なお、文字コードについては「キャラクタ・コード表」を参照してください。

画面の３行目に注目してください。配列の３番目の要素には数値０が格納されています。０はC言語では文字列の終わりを表すことを意味する大事な働きをします。０を特殊文字としてヌル文字と呼び、'¥０'と記述することがあります。したがって文字配列の要素数は、(格納する文字数＋１)以上でなければなりません。

5．2 関数

関数はCプログラミングの中で、最も基本的な実行単位です。関数は1つのまとまった仕事をしてくれる、独立した副プログラムです。作成された関数には名前がつけられ、新しい命令としてプログラムの中で使うことができます。
関数はユーザー（利用者）が作成する以外に、C言語のシステムによって提供されるライブラリ関数があります。ライブラリ関数は一般的に役に立つ関数を集めたもので、これまで使ってきたprintf ()、scanf () もその1つです。本機で用意してあるライブラリ関数については、「7．ライブラリ関数」を参照してください。
関数は互いに対等ですが、main () だけは特別な関数でプログラムの実行はこの関数から始まります。そのため主（メイン）関数と呼びます。Cプログラムはmain () を中心に、ユーザー関数、ライブラリ関数を組み合わせて作っていきます。

■関数を作る

関数は一般に、呼び出し側からデータを受け取ってなんらかの処理をした後、結果を返します。2つの整数の和を求める関数plus () を例に、関数の作りかた（定義）と使いかたを説明します。

```
int plus (int a, int b)
 {
  return (a + b) ;
 }
```

先頭行が関数の定義にとって重要な意味を持ちます。

```
int  plus (int a, int b)
 |    |     └── 引数の並び（引数がなければ空欄のまま）
 |    └── 関数の名前
 └── 関数の型
```

上の例では「関数の名前はplusで、2つのint型の値を受け取り、処理した結果をint型の値として返す」ことを表しています。なお、値を受け取る変数を引数（ひきすう）といい、関数の中では他の変数と全く同じ働きをします。関数の型は関数が返す値の型を意味します。plus () 関数は実行結果をint型データとして返します。
関数の内部での処理は { } で囲んだブロックの中に記述します。

```
 {
  return (a + b) ;
 }
```

return文は関数の実行を終わる働きをします。そのときreturnのすぐ後ろの値を関数の値（戻り値）として返します。plus () は (a + b) の値を返して実行を終わります。
なお、関数名についての規則は変数名のつけかたと同じです。

■関数の呼び出し

関数plus () を使ったプログラムの全体を示します。

```
10   main ()
20    {
30     int plus (int, int) ;
40     int x ;
50     x = plus (10, 20) ;
60     printf ("10+20=%d¥n", x) ;
70    }
80   /* カンスウ テイギ */
90   int plus (int a, int b)
100   {
110     return (a + b) ;
120   }
```


関数を呼び出す側では、あらかじめ呼び出す関数を宣言しておかなければなりません。行番号30が関数の宣言で関数定義の先頭行と同じ形式で宣言します。ただし、関数宣言では引数は型名だけでもかまいません。また宣言の最後にセミコロン ; が必要です。

行番号50で関数の呼び出しが行われ、2つの数値が渡されます。

```
呼び出し側   x = plus (10,      20) ;
     戻り値 ↑         ↓ 引数   ↓
関数側          plus (int a,   int b) ;
```

関数の戻り値は変数xに代入されます。関数自身が、変数のように値を持っている様子に注目してください。
呼び出す側から関数へ渡す値を実引数、関数側の引数を仮引数と呼びます。実引数は変数でもかまいません。お互いの引数はそれぞれ呼び出す側と呼び出される側の窓口です。関数内部で仮引数の値が変化しても実引数に影響を与えません。

■戻り値のない関数

次のmessage () のように戻り値を持たない関数をvoid型関数といい、関数名の前にキーワードvoidを記述します。

```
10   main ()
20    {
30     void message (void) ;
40     message () ;
50    }
60   /* void カンスウ */
70   void message (void)
80    {
90     printf ("void カンスウ") ;
100   }
```

message () は引数を持たない関数でもあります。この場合は、引数がないことをはっきりさせるため、関数の定義および宣言の引数並びにはvoidと記述します。ただし、関数呼び出し（行番号40）では引数並びは空欄にします。

■変数の有効範囲

関数の中で宣言された変数の有効範囲（代入したり読み出したりできる範囲）は関数内部に限られます。このような変数を局所（ローカル）変数といいます。

```
１０　　main ()
２０　　 {
３０　　  void　fun (void) ;
４０　　  int　i;
５０　　  for (i = ０ ; i < ３ ; i ++)　{
６０　　      printf ("main　i =%d　", i) ;
７０　　      fun () ;
８０　　  }
９０　　 }
１００　／*　カンスウ　*／
１１０　void　fun (void)
１２０　 {
１３０　      int　i = １０ ;
１４０　      printf ("　fun　i =%d ¥n", i) ;
１５０　}
```

プログラムを実行すると画面には次のように出力されます。

```
main　i = ０　fun　i = １０
main　i = １　fun　i = １０
main　i = ２　fun　i = １０
*EXIT　(９０)
```

このプログラムでは main () 関数と fun () 関数とで同じ名前の変数 i を使っていますが、それぞれの変数は互いに独立しています。main () 関数で i の値が変化しても、呼び出された fun () 関数の i は影響を受けていません。
局所変数は、変数宣言された関数の外から操作できません。また関数間で同じ名前を使っても、お互いが影響し合うことはありません。したがって関数ごとに独立してプログラミングできるため、プログラミング効率がよくなり、またプログラムのミス（バグ）も発生しにくくなります。

関数の外部で宣言される変数を大域（グローバル）変数といい、どの関数からでも共通して参照できます。

```
１０　　int　g = ０ ;
２０　　main ()
３０　　 {
４０　　  void　fun (void) ;
５０　　  int　i;
６０　　  for (i = ０ ; i < ３ ; i ++)　{
７０　　      printf ("g =%d ¥n", g) ;
８０　　      fun () ;
９０　　  }
１００　}
１１０　／*　カンスウ　*／
１２０　void　fun (void)
１３０　 {
１４０　  g ++ ;
１５０　}
```


プログラムを実行すると画面には次のように出力されます。

```
g = ０
g = １
g = ２
*EXIT　(１００)
```

変数 g は fun () 関数で変更することができ、main () から fun () が呼び出されるたびに g の値が１ずつ増加していきます。

大域変数は関数の間で共通の変数が必要な場合など、特別な場合にしか使いません。大域変数の多用は、プログラムをわかりにくくし、関数の独立性を損ないます。
Ｃプログラミングは必要な関数を作り、それを組み合わせて全体のプログラムを構成するというのが基本です。そのため使う変数はできるだけ局所的にすることで、それぞれの関数の独立性を保つことを心がけなければなりません。

■変数の存在期間

これまで使ってきた局所変数は、宣言した関数が呼び出されるたびに生成され、その実行が終わるたびに消滅します。このような変数を自動（auto）変数とも呼びます。
変数宣言で、キーワード **static** を使うとその変数は静的変数になります。

```
static　int　s ;
```

静的変数はプログラムの終了時まで存在します。関数内で宣言された静的変数は、その関数の実行が終了した後でも値を保持し続け、次に同じ関数が呼び出されるとその値が受け継がれます。

```
１０　　main ()
２０　　 {
３０　　  void　count (void) ;
４０　　  int　i;
５０　　  for (i = ０ ; i < ３ ; i ++)
６０　　      count () ;
７０　　}
８０　　／*　カンスウ　テイキ゛　*／
９０　　void　count (void)
１００　 {
１１０　  int　a = ０ ;
１２０　  static　int　s = ０ ;
１３０　  printf ("a =%d　s =%d ¥n", a, s) ;
１４０　  a ++ ; s ++ ;
１５０　}
```

画面には次のように表示されます。

```
a＝0　s＝0
a＝0　s＝1
a＝0　s＝2
＊EXIT　(70)
```

自動変数ａの値は関数呼び出しのたびに０に初期化されますが、静的変数ｓの値は前の内容に１ずつ加算された値になっています。

５．３　ポインタ

ポインタはポイントするもの、すなわち何かを指し示すものという意味です。プログラミングでは、データが記憶されている場所を指し示すことを意味します。最初はデータが記憶されているメモリのアドレス（番地）のことであると理解しておけばよいでしょう。
ポインタ（アドレス）を格納するためには特別な変数が必要となります。それをポインタ変数といいます。Ｃプログラミングの説明では、ポインタとポインタ変数はときどき区別されないで用いられることがありますので注意してください。

■変数のアドレスとポインタ

変数の宣言によって、変数にはメモリのある場所が割り当てられます。変数への代入とは、実際にはその場所にデータを記憶することにほかなりません。変数に割り当てられた記憶場所のアドレスはアドレス演算子 ＆ を使うことで得られます。たとえば変数ａのアドレスは次の式から得られます。

```
&a
```

ポインタ変数は他の変数と同じく、変数宣言しなければなりません。このとき、どのデータ型変数でアドレスを格納するかも明示する必要があります。たとえばint型の変数を扱うポインタ変数は次のように宣言します。

```
int  *p;
```

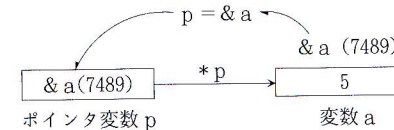
＊ はポインタ演算子と呼ばれ、ｐがポインタ変数であることを示します。また＊演算子はポインタの前につけることで、そのポインタの指し示すアドレスに格納されている値を返します。

```
10  main()
20  {
30    int  a=5;
40    int  *p;
50    p=&a;
60    printf("&a=%p:  a=%d¥n", &a, a);
70    printf("p=%p:  *p=%d¥n", p, *p);
80  }
```

プログラムが実行されると画面は次のようになります。

（注）７４８９（アドレス）はメモリが使用されている状態によって変わります。

```
&a=7489:  a=5
p=7489:  *p=5
*EXIT  (80)
```

行番号６０、７０のprintf文中の %p はポインタを出力するための変換指定です。出力画面から変数ａのアドレスが16進数で７４８９に割り当てられていることがわかります。また＊ｐがポインタの指し示す内容、すなわち変数ａの値を表していることも理解できます。

■配列とポインタ

ポインタは配列と密接な関係があります。ポインタを使って配列操作を行うと効率のよい、実行速度の速いプログラムを作ることができます。
配列要素はメモリ上の連続した領域に格納されます。またＣ言語では配列名は配列の先頭要素のアドレスを表します。文字配列str［ ］を例にとると、strと先頭要素のアドレス＆str［０］とは同じ意味になります。
ある配列要素を参照するには、先頭要素の添字を０としたときの要素の位置を添字として指示します。このような配列操作はポインタを使っても全く同じように行うことができます。

```
10  main()
20  {
30    int  i;
40    char  str[7]="string";
50    char  *ptr;
60    ptr=str;
70    for(i=0;i<7;i++)
80        printf("%c", *(ptr+i));
90  }
```

行番号６０で配列の先頭アドレスをポインタ変数ptrに代入しています。行番号８０では順次ポインタに加える値を増加させながら、その指し示す内容を出力させています。
配列要素とポインタとの関係を次の図に示します。配列の添字と、ポインタに加える値（オフセット値）が同じであることに注目してください。

str	's'	't'	'r'	・・・	'¥0'
	str[0]	str[1]	str[2]	・・・・・・	str[6]
	*ptr	*(ptr+1)	*(ptr+2)	・・・・・・	*(ptr+6)

インクリメント演算子を使うと、ポインタによる配列操作がスマートに行えます。

```
ptr++
```

たとえば、上記の80行を次のようにします。

```
80      printf("%c", *ptr++);
```

次のプログラムは与えられた文字列の長さを求めるもので、ポインタによる配列操作の典型的なスタイルといえます。

```
10  main()
20  {
30    char  str[10]="pointer";
```

```
40      char  *ptr;
50      int  n = 0;
60      ptr = str;
70      while (*ptr != 0)  {
80          ptr++;
90          n++;
100     }
110     printf("ﾓｼﾞﾚﾂﾉ ﾅｶﾞｻﾊ  %d¥n", n);
120  }
```

行番号７０の繰り返しの条件式は、ポインタの指し示す内容が０、すなわち、文字列の終わりであるかどうかを調べます。

行番号７０から１００までのリストは次のように簡潔に表示できます。

```
while (*ptr++  != 0)
    n++;
```

条件式は＊ptr の値を調べた後、ポインタ変数 ptr の値を１つ増加します。つまり、条件のチェックと行番号８０のポインタ操作を同時に行っています。

６．Ｃ言語の要点

６．１　本機のＣ言語仕様

本機のＣ言語の仕様は標準的なＣ言語に合わせるようにしていますが、次の項目は例外となります。

① 型定義（typedef）宣言できない。

② 列挙型（enum）宣言できない。

③ volatile 指定できない。

④ ビットフィールド宣言できない。

⑤ 配列型の auto 変数を初期化できる。（その場合は配列サイズを明示しなければならない。）

⑥ 構造（struct）型／共用（union）型は、引数や関数値にできない。

⑦ 構造（struct）型／共用（union）型は、条件演算子 a？b：c で使用できない。

⑧ ブロック（{}）内の宣言は、トップレベルのブロックでしかできない。

⑨ プリプロセッサの機能は、引数なしの＃define、＃include、＃if〜＃elif〜＃endif、＃ifdef〜＃elif〜＃endif である。

⑩ ライブラリ名・ストリーム名と同名の関数名・変数名は宣言できない。

⑪ main() は引数を持たない。

⑫ 定数式では次の演算子のみ有効である。

整数演算	単項－	単項＋	～						
	＊	／	％	＋	－	<<	>>	&	^ |
実数演算	単項－	単項＋							

（注）定数式では、整数と実数間の型変換はできません。

⑬ Ｃ領域以外のメモリへの書き込みは禁止されている。

⑭ ポインタでの関数呼び出しで、Ｃ領域以外のポインタ値は許されていない。

⑮ 秒記号（″）は扱えない。

⑯ 以下の制限がある。

＃define 文において	
個数	３２
名前の文字数	３１
本体の文字数	３１
名前の本体の保存バッファ	２５５
展開バッファ	８０
＃include 文において	
入れ子の深さ	２
＃if、＃ifdef 文において	
入れ子の深さ	２
定数の数字の数	７０
名前の文字数	３１
文字列の文字数	８０
型修飾（＊，[ ]，( )）	８
１関数内の goto ラベル	８
最外側の switch 文内の case ラベル	３２
構造体、共用体、配列、関数での宣言の深さ	４
構造体、配列での初期化変数の深さ	８
プロトタイプの深さ	３
最外側 () 内のプロトタイプ内の宣言数	３２
次の構文での入れ子の深さ（if，while，do，for，switch，{}）	１５
式の演算子の深さ	１６
式の演算子のオペランドの深さ	３２
式の深さ（tree 形式で）	３２

６．２　キーワード

次の識別子（名前）はＣ言語のキーワード（予約語）として使われるので、別の用途（変数名、関数名など）に使えません。

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

また、次のキーワードも本機では使用できません。

enum　typedef　volatile

６．３　定数

定数には整数定数、文字定数、文字列定数、実数定数、列挙定数（本機では使えない）があります。

表記法		表記例	備考
整数	１０進	１２８	――
	１６進	０ｘ２ｆ	先頭に ０x または ０X をつける
	８進	０５６	先頭に ０（ゼロ）をつける
実数		１．４９６，２．３４e７	e または E をつけて指数表記もできる
文字		‘Ａ’，‘¥ｎ’	シングルクォーテーションで１文字を囲む
文字列		"Programming"	ダブルクォーテーションで文字の並びを囲む

整数定数では数値の後ろに l または L をつけると、long（32ビット）として扱われます。それ以外は大きさに合わせて int または long です。

〈例〉　２６３４８Ｌ

実数定数では数値の後ろに f または F をつけると、float（32ビット）として扱われます。それ以外では double です。

〈例〉　２．６５ｆ

文字定数は int（16ビット）で表現され、ＡＳＣＩＩコードが使用されています。

Ｃ言語では次の表のエスケープ列を使った制御文字を出力することができます。エスケープ列は文字定数として扱われます。

制御文字	値（１６進）	機能
¥b	０ｘ０８	復帰（カーソルを１文字前へ移動）
¥n	０ｘ０Ａ	復帰改行（カーソルを次の行の先頭へ移動）
¥r	０ｘ０Ｄ	復帰（カーソルをその行の先頭へ移動）
¥t	０ｘ０９	水平タブ（カーソルを次の水平タブへ移動）
¥¥	０ｘ５Ｃ	文字¥を出力
¥’	０ｘ２Ｃ	文字’を出力
¥"	０ｘ２２	文字"を出力
¥?	０ｘ３Ｆ	文字？を出力
¥ddd		８進数 ddd（３桁）に対応する文字を出力
¥xhh		１６進数 hh に対応する文字を出力

６．４　扱える数値の範囲

整数型で扱える数値の範囲は次のとおりです。

データ型	扱う数値の範囲	符号なし（unsigned）	扱う数値の範囲
char	−128〜＋127	unsigned　char	0〜255
short	−32768〜＋32767	unsigned　short	0〜65535
int	−32768〜＋32767	unsigned　int	0〜65535
long	−2147483648〜＋2147483647	unsigned　long	0〜4294967295

実数型で扱える数値の範囲は次のとおりです。

データ型	扱う数値の範囲
float	±１e −99〜±9.999 e ＋99
double	±１e −99〜±9.999999999 e ＋99
long double	±１e −99〜±9.999999999 e ＋99

６．５　式と文

Ｃプログラミングでは式と文の違いを理解しておくことが大切です。式は演算の結果としての値を持ちます。式の最後にセミコロン ; をつけたものが文で、実行の基本単位となります。式が値を持つため、次のような文が許されます。

ａ＝ｘ＝５；

この文ではまず右の代入式（ｘ＝５）が評価された後、式の値である５がａに代入されます。

６．６　演算子

Ｃ言語には豊富な演算子が用意されています。演算子の意味と働きを説明します。

①算術演算子

演算子	意味	使用例	意味
＋	加算	ａ＋ｂ	ａにｂを加える
−	減算	ａ−ｂ	ａからｂを引く
＊	乗算	ａ＊ｂ	ａにｂを掛ける
／	除算	ａ／ｂ	ａをｂで割る
％	剰余	ａ％ｂ	ａをｂで割った余り

②代入演算子

演算子	使用例	意味	算術演算子による表記
＝	ａ＝ｂ	ａにｂを代入	――
＋＝	ａ＋＝ｂ	ａにｂを加えた結果をａに代入	ａ＝ａ＋ｂ
−＝	ａ−＝ｂ	ａからｂを引いた結果をａに代入	ａ＝ａ−ｂ
＊＝	ａ＊＝ｂ	ａにｂを掛けた結果をａに代入	ａ＝ａ＊ｂ
／＝	ａ／＝ｂ	ａをｂで割った結果をａに代入	ａ＝ａ／ｂ
％＝	ａ％＝ｂ	ａをｂで割った余りをａに代入	ａ＝ａ％ｂ

表以外にも⑥に示すビット演算子（～は除く）や⑦に示すシフト演算子による代入演算子も使えます。
（注）代入演算子の右辺と左辺のデータ型が異なる場合、一般的なC言語と結果が異なる場合があります。
型が異なる場合は算術演算子による表記を行ってください。

③インクリメント演算子とデクリメント演算子

演算子	使　用　例	意　　味	算術演算子による表記
＋＋	ａ＋＋または＋＋ａ	ａに１を加える	ａ＝ａ＋１
－－	ａ－－または－－ａ	ａから１を引く	ａ＝ａ－１

＋＋ａ（または－－ａ）を前置演算子、ａ＋＋（またはａ－－）を後置演算子といいます。次のように動作が異なりますから注意が必要です。

ｂ＝＋＋ａ；　ａに１を加えてからｂに代入する
ｂ＝ａ＋＋；　ｂにａを代入してからａに１を加える

④関係演算子

関係演算子は演算の結果が真のときは数値１、偽のときは０になります。

演算子	使用例	演算結果が真になる場合
＝＝	ａ＝＝ｂ	ａはｂに等しい
！＝	ａ！＝ｂ	ａはｂに等しくない
＜	ａ＜ｂ	ａはｂより小さい
＞	ａ＞ｂ	ａはｂより大きい
＜＝	ａ＜＝ｂ	ａはｂ以下である
＞＝	ａ＞＝ｂ	ａはｂ以上である

⑤論理演算子

論理演算子は結果が真のときは数値１、偽のときは０になります。

演算子	使用例	演算結果が真になる場合
＆＆	ａ＆＆ｂ	ａとｂがともに真である
¦¦	ａ¦¦ｂ	ａまたはｂが真であるか、ともに真である
！	！ａ	ａが偽である

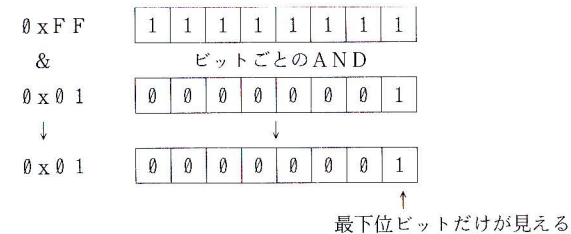
（注）¦¦は一般的なC言語では｜｜になります。

⑥ビット演算子

コンピュータが扱うデータの最小単位はビットです。C言語にはビット単位で演算を行うビット演算子が用意されています。

演算子	使用例	意　　味	演　算　の　目　的
＆	ａ＆ｂ	ビットごとのＡＮＤ	特定のビットのマスク
¦	ａ¦ｂ	ビットごとのＯＲ	特定のビットを１にセット
＾	ａ＾ｂ	ビットごとの排他的ＯＲ	特定のビットを反転
～	～ａ	ビットごとのＮＯＴ	全ビットの反転

（注）¦は一般的なC言語では｜になります。

＆演算子を使うと特定のビットを調べることができます。

⑦シフト演算子

各ビットを左、または右へシフトするシフト演算子が用意されています。

演算子	使用例	意　　味
＜＜	ａ＜＜ｂ	ａをｂビット左へシフト
＞＞	ａ＞＞ｂ	ａをｂビット右へシフト

右へシフトする場合、符号付き整数型は空いたビットが符号ビットで埋められます。符号なしの場合は０で埋められます。

⑧型変換とキャスト演算子

異なるデータ型どうしの演算や、代入演算、また関数の引数および戻り値に対して自動的な型変換が発生します。

１．異なる型どうしの演算は、次の優先順位（ビット長に準ずる）に従って変換されてから演算が行われます。

char < int < long < float < double

２．代入によって、右辺の値は左辺の型に変換されます。このとき、順位の高い型から低い型への変換で丸めや切り捨てが行われるので、注意が必要です。
関数の戻り値は、代入と同様に、関数の型に変換されます。

３．キャスト演算子を使うと型変換を明示的に行うことができます。

（データ型）式

式の値を（　）で指定した型に強制的に変換します。

```
〈例〉int    a；
     double  b, c；
     ・・・・・・
     c = b／(double) a；
     ・・・・・・
```

⑨いろいろな演算子

■アドレス演算子とポインタ演算子

Ｃ言語にはアドレスを直接操作する演算子があります。

＆　：変数および配列要素のアドレスを取り出す（アドレス演算子）

＊　：ポインタが指し示す変数への参照（ポインタ演算子）

■条件演算子

条件演算子は if 文と同じような働きをする演算子です。

ａ？ｂ：ｃ

条件式ａが真なら式ｂを、偽なら式ｃを評価した値を返します。

ａ＝ｘ＞＝０？１：０

ｘが０または正ならば１が、負であれば０がａに代入されます。

■ sizeof 演算子

オブジェクトを格納するのに必要なバイト数を求める演算子です。

sizeof（型）　：データ型の大きさをバイト数で返す

sizeof　式　　：式の値を格納するのに必要なバイト数を返す

たとえば次の例ではｓに２（バイト）が代入されます。

ｓ＝sizeof（int）；

⑩演算子の優先順位と結合規則

Ｃプログラミングでは演算子は数や機能が豊富なため、使いかたに慣れるのは大変です。特に演算子を組み合わせた式では、演算子の優先度に注意を払わなければなりません。

表に演算子の優先順位と結合規則を示します。

優先度	演算子の種類	結合規則	演算子
高い	１次式演算子	左から右	（　）［　］－＞
	単項演算子	右から左	！　～　＋＋　－－　＋　－　（型）　＊　＆　sizeof
	２項演算子	左から右	＊　／　％
		左から右	＋　－
		左から右	＜＜　＞＞
		左から右	＜　＜＝　＞　＞＝
		左から右	＝＝　！＝
		左から右	＆
		左から右	＾
		左から右	｜
		左から右	＆＆
		左から右	｜｜
	条件演算子	右から左	？：
	代入演算子	右から左	＝　＋＝　－＝　その他
低い	コンマ演算子	左から右	，

表において、同じ行の演算子の優先度は同じです。結合の方向とは同じ優先度の演算子が続いたときに、式の評価の方向をいいます。次の式は右の式のように評価されます。

ａ＋ｂ－ｃ　－－－－＞　（ａ＋ｂ）－ｃ　　：左から右へ結合

ａ＝ｂ＋＝ｃ　－－－－＞　ａ＝（ｂ＋＝ｃ）　：右から左へ結合

６．７　いろいろな構文

Ｃプログラムでの実行の流れは次の制御構造に分類できます。

逐次構造　：並べられた順に、文を実行する。

選択構造　：条件によって、実行の流れを分岐する。（条件文）

ループ構造：ある一定の条件下で、実行を繰り返す。（繰り返し文）

①単文と複文

ほとんどの文は式の文で、次のように文の最後はセミコロン　；　で終わります。

式；

複数の文を１つの文として扱えるようにするために複文が用意されています。複文は次のように単文をカッコ　｛｝　で囲みます。複文はブロックともいいます。

```
{
  文
  ・・・・
  文
}
```

一般にはブロック内での変数宣言ができますが、本機ではトップレベルのブロックでしか許されていません。

②選択文

Ｃ言語では if ～ else 文、switch ～ case 文の選択文が用意されています。

■ if ～ else 文

if ～ else 文には次のような形式があります。

１．if（式）　　式が真なら文を実行し、偽なら単に通り抜ける。
```
    文
```

２．if（式）　　式が真なら文１を、偽なら文２を実行する。
```
    文１
  else
    文２
```

３．if（式１）　　式１が真なら文１を、そうでなく式２が真なら文２を、式１、式２とも偽なら文３を実行する。
```
    文１
  else  if（式２）
    文２
  else
    文３
```

■ switch ～ case 文

switch ～ case 文は次の形式になっています。

```
switch　（条件式）　｛
      case　定数式１：文１
                      break；
      case　定数式２：文２
                      break；
          ・・・・・・
      case　定数式ｎ：文ｎ
                      break；
      default：文
｝
```

switch 文は最初に条件式を評価し、その値と一致する定数式（ラベル）の文を実行します。

実行した後、break 文で switch 文を抜けます。break 文がないとその後に続くラベルの文へ進みます。

条件式に一致する定数式がない場合はキーワード default に続く文を実行します。

switch ～ case 文の条件式、定数式の型は整数型でなければなりません。

また、評価は unsigned　int 型に変換して行われます。

③繰り返し文

Ｃ言語では for 文、while 文、do ～ while 文が用意されています。

■ for 文

for 文は次の形式になります。

```
for（式１；式２；式３）
    文
```

for 文は最初に１回だけ式１を評価します。式２が真であれば文を実行した後、式３を実行します。同様の動作を式２が偽になるまで繰り返します。

■ while 文

while 文は次の形式になります。

```
while　（条件式）
    文
```

条件式を評価し、真である間、文を繰り返し実行します。

■ do ～ while 文

do ～ while 文は次の形式になります。

```
do
    文
while　（条件式）；
```

文を実行した後、条件式を評価します。条件式が偽になるまで文を繰り返し実行します。

④ジャンプ文

ジャンプ文はプログラムの実行の流れを無条件に移す働きをします。ジャンプ文として goto 文、continue 文、return 文があります。

■ goto 文

goto 文は次のように使います。

```
goto　ラベル；
  ・・・・・・
ラベル：　文
```

実行の流れが無条件にラベルの位置にジャンプします。ジャンプする先は同じ関数の中に限定されます。

■ continue 文

continue 文は繰り返し文の中にのみ置くことができます。continue 文は繰り返し文で、ムダな実行を省略して実行速度を上げるためなどに使います。

```
・・・・・・
for（i＝０；i＜１００；i＋＋）　｛
      ・・・・・・
  if（i％２＝＝０）
      continue；
  printf（"％d￥n"，i）；
｝
  ・・・・・・
```

上の例では i が奇数のときだけ数値が出力されることになります。

■ break 文

繰り返し文や switch ～ case 文で、構文を抜け出し１つ外側の実行単位に移るために使います。

```
・・・・・・
for（i＝０；i＜１００；i＋＋）　｛
      if（a［i］＜０）
        break；
・・・・・
｝
  ・・・・・
```

上の例では配列の要素が負になったら for 文を抜けます。

■ return 文

return 文は実行中の関数を抜け、その関数を呼び出した関数に実行を戻します。次のように戻り値がある場合は return 文の後ろに記述します。

```
return（式）；
```

（　）は省略できますが、つけておいたほうがわかりやすくなります。戻り値がない場合に（　）をつけるとエラーになります。

6．8　記憶クラス

C言語では、変数をメモリのどの位置に配置するかによって、自動的と静的の２つの記憶クラスに分かれます。またプログラムで宣言された変数の場所が、関数またはブロックの中か外かによって、有効範囲が決められます。

種　類	有　効　範　囲	存　在　時　間
auto	宣言された実行単位の中で有効。 ただし、本機ではトップレベル以外のブロックでの宣言はできない。	実行単位から抜けた後は消滅する。
register	auto 変数と同じであるが、実行速度を上げるために用いる。ただし本機では auto とみなされる。	auto と同じ。
static	宣言された実行単位の中で有効。 ただし、本機ではトップレベル以外のブロックでの宣言はできない。	プログラム実行中は消滅せず、値を保存する。
extern	宣言された実行単位で有効。 ただし、本機では「記憶クラス省略」とみなされる。	プログラム実行中は消滅せず、値を保存する。 ただし、本機では関数内で宣言すると auto 、関数外では static と同じ。

6．9　多次元配列

配列の配列を２次元配列、またその配列を３次元配列といいます。C言語ではこのように多次元配列が使えます。本機では８次元までの配列が使えます。

２次元配列の宣言は次の例のように、配列の大きさを［　］で囲んで並べます。

```
char　color［3］［6］
             ↑    ↑
             行   列
```

１次元配列と同様、宣言のときに初期化ができます。

```
char　color［3］［6］＝｛"white"，"red"，"blue"｝；
```

図のように配列要素にデータが格納されます。

6．10　構造体

配列は同じ型のデータを扱うのに便利なデータ型であるのに対して、構造体、共用体は性質の異なるデータの集まりを、ひとつのデータ構造として表現するためのデータ型です。

たとえば野球選手の打撃成績を表現するデータ型は次のようになります。

```
struct　batting　｛
        char　＊name；
        float　ave；
        int　　homer；
｝；
```

struct は構造体であることを示すキーワードです。**batting** は構造体のタグといい、定義される構造体の形式につける名前です。構造体の要素は ｛ ｝ の中に記述します。選手の名前を＊name、打率を ave 、ホームラン数を homer の各要素に格納します。
構造体型の変数は次のように宣言します。

```
struct　batting　batter；
```

タグ名を省略して直接、構造体変数を宣言できます。

```
struct　｛
    char　＊name；
    float　ave；
    int　　homer；
｝　batter；
```

各要素への参照は次のように、変数名と要素名を ． でつないで表記します。

```
変数名．要素名
```

選手名、打率、ホームラン数を代入してみましょう。

```
batter．name＝"ナカハラ"；
batter．ave＝3．14；
batter．homer＝40；
```

6．11　プリプロセッサ

C言語の処理系ではソースプログラムの中にプロセッサ指令があると、プログラムをコンパイルする前に指令に従った「前処理」を行います。いくつかのプリプロセッサのうち、「文字列定義／マクロ定義」を行う＃define、「ファイルの取り込み」を行う＃include と「条件コンパイル」を行う＃if、＃ifdef、＃ifndef を説明します。

■ ＃define 指令

＃define は指定された文字列の置き換えを行います。

```
＃define　文字列１　文字列２
```

文字列２は文字列１に置き換えられてからコンパイルされます。＃define は次の例のように、プログラム中で使われる定数を文字列に置き換える場合などによく使われます。

```
＃define　PI　3．141592
```

文字列ＰＩはプログラムの中では円周率を意味する定数として使うことができます。

置き換え

ｓ＝ＰＩ*ｒ*ｒ；　➡　ｓ＝３．１４１５９２*ｒ*ｒ；

＃define 指令では次のことに注意が必要です。

● 文字列１は大文字で書くことが習慣となっている。

● 構文の途中で改行はできない。

本機ではあらかじめ、次のように各文字列が定義されています。

```
＃define　ＮＵＬＬ　　０
＃define　ＥＯＦ　　　－１
＃define　ＦＩＬＥ　　int
```

また本機では次のような引数付きマクロを定義できません。

```
＃define　ＳＱＲ（ｘ）　（（ｘ）＊（ｘ））
```

■＃include 指令

＃include 指令は指定したファイルを、ソースプログラムのその位置に取り込みます。一般のＣ言語処理系では、ライブラリ関数を使うためのヘッダファイルを読み込むためによく使われますが、本機ではヘッダファイルの読み込みは不要です。

ファイルの読み込みは次のようにします。

```
＃include "ファイル名"
```

■＃if～＃elif～＃else～＃endif 指令

条件文により、コンパイルするテキストを指定します。

```
＃if　条件式
　　　テキスト
［＃elif　条件式
　　　テキスト］
［＃else
　　　テキスト］
＃endif
```

条件式を満たすところに記述されているテキストのみをコンパイルします。なお、条件式を満たすところがなければ、次の＃elif の条件式を判別します。＃elif、＃else は省略できます。

■＃ifdef、＃ifndef 指令

```
＃ifdef　文字列
　　　テキスト
［＃else
　　　テキスト］
＃endif
```

文字列が＃define 指令で指定されているかどうかにより、テキストをコンパイルするかしないかを指定します。

＃ifdef は文字列が指定されているときにコンパイルを行い、＃ifndef は文字列が指定されていないときにコンパイルを行います。

7．ライブラリ関数

本機で用意されているライブラリ関数を説明します。本機ではライブラリ関数のためのヘッダファイルは必要ありません。

本機では、標準入力装置・出力装置（stream）は次のように、各機器に割り当てられています。

入力　stream：stdin（キーボード）

出力　stream：stdout（画面）、または stdprn（プリンタ）

ＳＩＯ　stream：stdaux（11ピン　半二重通信）

　　　stream：stdaux１（11ピン　全二重通信）

また、あらかじめ次のように＃define されています。

```
＃define　ＮＵＬＬ　　０
＃define　ＥＯＦ　　　－１
＃define　ＦＩＬＥ　　int
```

本機では入力のためのライブラリ関数において、SHIFT（または 2ndF）＋ ↵ の入力は、それぞれ次の値を返します。

getch　　　　　：０ｘＦＦ

その他の入力関数：ＥＯＦ

ラムデータファイル、ＳＩＯの入力関数では次のコードを区切り文字とします。

行の区切り：0x0d、0x0aまたは0x0d+0x0a

ファイルの終了：0x1a

（0x0d+0x0aは入力時に0x0aに変換されます。）

（注）区切りコードはできるだけ0x0aを使用してください。

ラムデータファイル、ＳＩＯの出力関数では次のコードを区切り文字とします。

行の区切り：ヌル

（0x0aは出力時に0x0d+0x0aに変換されます。）

関数形式、説明文の記述において、［　］で囲まれた記述はオプションで、記述の省略が可能です。

7．1　標準入出力関数

● getc，getchar，fgetc

形式：

```
int　getc（ＦＩＬＥ＊　stream）；
int　getchar（void）；
int　fgetc（ＦＩＬＥ＊　stream）；
```

機能： １文字を読み込みます。stream が stdin のときは ↵ を押すと文字を読み込みます。

getc　　　入力 stream から読み込みます。

getchar　stdin から読み込みます。

fgetc　　入力 stream から読み込みます。

戻り値： 読み込んだ文字を返します。

● gets, fgets

形式： char＊　gets（char＊　s）；

char＊　fgets（char＊　s，int　n，ＦＩＬＥ＊　stream）；

機能： 文字列を読み込みます。文字列をｓに読み込むとき、オーバーフローチェックは行いません。

gets　stdin から ↵ の入力で終わる文字列をｓに読み込みます。文字列の終わりの ↵ はヌル文字'¥0'に置き換えられます。

fgets　入力 stream から文字列をｓに読み込みます。読み込みは、stdin では（n－1）個の文字を読むか、↵ を読み込むことで終わります。それ以外の stream では、（n－1）個の文字を読むか、行の区切り文字を読み込むことで終わります。文字列の最後（↵ を入力したときはその後）にヌル文字'¥0'が付加されます。

戻り値： 読み込んだ文字列ｓを返します。ＥＯＦのときはヌルを返します。

● scanf, fscanf, sscanf

形式： int　scanf（const　char＊　format〔，address，．．．〕）；

int　fscanf（ＦＩＬＥ＊　stream，const　char＊　format〔，address，．．．〕）；

int　sscanf（char＊　s，const　char＊　format〔，address，．．．〕）；

機能： 書式付き入力を行います。

scanf　stdin から読み込みます。

fscanf　入力 stream から読み込みます。

sscanf　文字列ｓから読み込みます。

読み込まれた文字の並びを、書式を与える文字列 format で指示されたデータ型に変換し、対応する引数 address，．．．の示すアドレスに格納します。

書式で与えられた変換指定の数と、引数で示すアドレスの数は同じでなければなりません。多すぎた場合は無視されますが、不足した場合、結果は保証されません。

scanf と fscanf は ↵ を押すと入力されます。sscanf はヌル文字'¥0'が文字列の終了とみなされます。

scanf、fscanf は次の場合に処理を終了します。

1）フォーマット終了（文字列終了マーク'¥0'の読み込み）
2）読み込みデータとフォーマットの不一致
3）読み込み終了

戻り値： 正しく変換、入力したデータの数を返します。入力データがない場合は０を返します。ＥＯＦのとき、あるいはエラーのときは－１を返します。

■入力の書式指定

入力データの書式を与える文字列は、書式指定、空白文字、非空白文字で構成されます。空白文字は空白（SPACE）、タブ（TAB）、↵ です。非空白文字は空白文字、％文字以外の全ての文字です。

scanf（）は次のように動作します。

- 書式文字列中に空白文字があると、入力における空白文字を、次に非空白文字が現われるまで読みとばします。

- 書式文字列中に非空白文字があると、それに一致する文字を読み込みます。不一致であれば、不一致文字は読み込まないで終了します。
- 書式指定は入力フィールドを読んで指定に従って変換し、引数で指定されるアドレスへ格納します。

たとえばコンマ，を区切りとするには次のように書式指定します。

scanf（"％d，％d，％d"，＆a，＆b，＆c）；

入力データの形式は次のようになります。

１２，３４，５６ ↵

◆変換指定

入力された文字の並びをどのようなデータ型として解釈するか、つまりどのように変換するかを指示するのが変換指定です。変換指定は％で始まり変換文字で終わる綴りです。

変換指定	期待される入力の形式	変換の形式
％d	10進整数形式の文字列	int に変換
％i	10／８／16進整数形式の文字列	〃
％o	８進整数形式の文字列	〃
％u	符号なし10進整数形式の文字列	unsigned int に変換
％x	16進整数形式の文字列	int に変換
％f	浮動小数点数形式の文字列	float に変換
％e	〃	〃
％g	〃	〃
％c	１文字がそのまま入力となる	char に変換
％s	文字列がそのまま入力となる	文字列の最後に'¥0'がつく
％p	16進４文字の文字列（89ab など）	ポインタに変換

◆変換のオプション指示

％記号と変換文字の間にオプションの指示を与えることができます。

％〔＊〕〔入力幅〕〔変換文字の修飾〕変換文字

〔　〕で囲んだ各要素がオプション指示で、必要に応じて指定します。

①代入抑制文字

＊　変換指示に対応するフィールドは読みとばされ、代入されません。

②入力幅

n　変換する文字数を指定します。
指定したフィールド幅の中に空白文字があった場合、空白文字までが入力されます。

省略　空白文字、または変換できない文字までが入力幅となります。

③変換文字の修飾

l（エル）　変換文字が整数への変換を指示している場合、long　int に変換します。
変換文字が実数への変換を指示している場合、double に変換します。

L　変換文字が実数への変換を指示している場合、long　double に変換します。

● putc, putchar, fputc

形式：　int　putc（int　c，ＦＩＬＥ＊　stream）；
　　　　int　putchar（int　c）；
　　　　int　fputc（int　c，ＦＩＬＥ＊　stream）；

機能：　１文字を出力します。
　　putc　　　出力 stream に１文字を出力します。
　　putchar　stdout に１文字を出力します。
　　fputc　　出力 stream に１文字を出力します。

戻り値：　書き込まれた文字を返します。エラーが発生した場合は、ＥＯＦを返します。

● puts, fputs

形式：　int　puts（const　char＊　s）；
　　　　int　fputs（const　char＊　s，ＦＩＬＥ＊　stream）；

機能：　文字列を出力します。
　　puts　ヌル文字'￥0'で終わる文字列ｓを、ヌル文字を改行文字に置き換えて stdout に出力します。
　　fputs　ヌル文字'￥0'で終わる文字列ｓを出力 stream に出力します。改行文字はつけず、最後のヌル文字は出力されません。

戻り値：　負でない値を返します。エラーが発生した場合は、負の値を返します。

● printf, fprintf, sprintf

形式：　int　printf（const　char＊　format〔，arg，．．．〕）；
　　　　int　fprintf（ＦＩＬＥ＊　stream，const　char＊　format〔，arg，．．．〕）；
　　　　int　sprintf（char＊　s，const　char＊　format〔，arg，．．．〕）；

機能：　引数 arg，．．．によって与えられる数値や文字、文字列を、書式文字列 format に従って変換された文字の並びとして、出力します。
format の中の文字はそのまま出力されますが、％で始まる書式制御文字列は、それに対応する引数の書式制御に使用されます。format で指定した書式制御文字列の型と引数の型が対応していなかったり、不足している場合は、出力は保証されません。引数の数が書式文字列より多い場合には、余った引数は無視されます。
　　printf　　stdout に出力します。
　　fprintf　出力 stream に出力します。
　　sprintf　文字列ｓに書き込みます。最後にヌル文字'￥0'を付加します。

戻り値：　出力した文字数を返します。エラーが発生した場合は、負の数が返されます。

■出力の書式指定

printf は書式文字列に従って処理されます。書式文字列中に変換指示があると、対応する引数を指定に従って変換して出力します。変換指定でない文字列はそのまま出力されます。

◆変換指定

変換指定	引数の種類	変換の形式
％d	int	符号付き10進数で表示
％i	int	符号付き10進数で表示
％o	int	符号なし８進数で表示
％u	int	符号なし10進数で表示
％x	int	符号なし16進数で表示（abcdef を使う）
％X	int	〃　　（ＡＢＣＤＥＦを使う）
％f	double	〔－〕ddd. ddd の形の10進数で表示（ｄは10進数１桁）
％e	double	〔－〕d. ddde ± dd の指数形式で表示
％E	double	〔－〕d. dddＥ± dd の指数形式で表示
％g	double	上記のｆまたはｅのうち、短い方で表示
％G	double	上記のｆまたはＥのうち、短い方で表示
％c	int	１文字として表示（int は unsigned　char に変換）
％s	文字列	引数で指し示される文字列を表示
％p	ポインタ	引数をポインタとして表示

◆変換指定のオプション指示

変換指定は％で始まり、変換文字で終わる綴りですが、％と変換文字の間にはオプション指示を与えることができます。変換指示子の一般的な形式は次のようになります。

　％〔フラグ〕〔印字幅〕〔．精度〕〔サイズ〕変換文字

〔　〕で囲んだ各要素がオプション指示で、必要に応じて使うことになります。また記述する順序も上の形式に従わなければなりません。それぞれのオプションを説明しましょう。

①フラグ
　－　変換の結果は左詰めで表示されます。
　＋　符号付きは必ず＋または－符号から始まります。
　＃　変換文字がｏの場合は先頭が０（ゼロ）で始まります。
　　　ｘ（またはＸ）の場合は０ｘ（または０Ｘ）で始まります。
　０　数値型文字のフィールド内をスペースの代わりに０で埋めます。ただし、ｄ、ｉ、ｏ、ｕ、ｘ、Ｘで精度指定がある場合は、０フラグは無効です。０と－の両方があれば、－の処理になります。
　省略　右詰めで出力されます。

②印字幅
　ｎ　変換された結果を表示する桁数（フィールド幅）の指定です。
　　　出力の幅がフィールド幅より少ない場合、残りは空白で埋められます。
　　　たとえば％１０ｄと指定すると10文字分の幅で１０進数として出力されます。
　０ｎ　ｎと同じくフィールド幅を指定します。
　　　出力の値がｎより小さい場合は先頭から０で埋められます。
　省略　必要なフィールド幅がとられます。

③精度指定
　ｎ　浮動小数点数の出力で小数点以下の桁数を指定します。（最後の桁は丸められる）
　省略　変換文字がｅ、Ｅ、ｆでは小数点以下は６桁。
　　　変換文字がｇ、Ｇでは有効数字がすべて出力されます。

④サイズ

l（エル）　l（エル）指示でd、i、o、u、x、Xはlong型、nはlong型へのポインタになります。

● fflush

形式：　int　fflush（ＦＩＬＥ＊　stream）；

機能：　streamのバッファをフラッシュします。

入力streamの場合はバッファの内容がクリアされます。その他は何もしません。streamがヌルのときは、全streamに対してフラッシュします。

戻り値：　0を返します。

● clearerr

形式：　void　clearerr（ＦＩＬＥ＊　stream）；

機能：　streamに関するエラーとＥＯＦフラグをクリアします。

戻り値：　ありません。

７．２　文字処理関数

● isalnum，isalpha，iscntrl，isdigit，isgraph，islower，isprint，ispunct，isspace，isupper，isxdigit

形式：

```
int　isalnum（int　c）；
int　isalpha（int　c）；
int　iscntrl（int　c）；
int　isdigit（int　c）；
int　isgraph（int　c）；
int　islower（int　c）；
int　isprint（int　c）；
int　ispunct（int　c）；
int　isspace（int　c）；
int　isupper（int　c）；
int　isxdigit（int　c）；
```

機能：　文字ｃの分類をします。

isalnum　英字であるか、または数字であるかを判定します。

isalpha　英字であるかを判定します。

iscntrl　制御文字であるかを判定します。
（0x00～0x1f、0x7fを真と判定）

isdigit　数字であるかを判定します。

isgraph　図形（グラフ）文字かを判定します。
（0x21～0x5f、0x61～0x7e、0x81～0x9f、0xa1～0xf7を真と判定）

islower　英小文字であるかを判定します。

isprint　印字可能な文字であるかを判定します。
（0x20～0x7e、0x80～0xf7を真と判定）

ispunct　区切り文字であるかを判定します。
（0x21～0x2f、0x3a～0x40、0x5b～0x5f、0x7b～0x7e、0x81～0x9f、0xa1～0xf7を真と判定）

isspace　空白文字（空白、タブ、復帰、改行など）であるかを判定します。
（0x09～0x0d、0x20を真と判定）

isupper　英大文字であるかを判定します。

isxdigit　16進法の表記文字であるかを判定します。

戻り値：　判定が真の場合は0以外の値を返します。偽の場合は0を返します。

● tolower，toupper

形式：

```
int　tolower（int　c）；
int　toupper（int　c）；
```

機能：　tolower　英文字を小文字に変換します。

toupper　英文字を大文字に変換します。

戻り値：　変換した文字を返します。

７．３　文字列処理関数

● strcat

形式：　char＊　strcat（char＊　ｓ１，const　char＊　ｓ２）；

機能：　文字列ｓ２を文字列ｓ１の最後につけ加えます。付加したときオーバーフローチェックは行いません。

戻り値：　連結された文字列へのポインタを返します。

● strchr

形式：　char＊　strchr（const　char＊　s，int　c）；

機能：　文字列ｓを先頭から調べて、文字ｃを探します。

戻り値：　最初に見つかった文字ｃへのポインタを返します。

見つからなかったときはＮＵＬＬポインタを返します。

● strcmp

形式：　int　strcmp（const　char＊　ｓ１，const　char＊　ｓ２）；

機能：　文字列ｓ１とｓ２を比較します。各文字列の先頭から順に、異なる文字が現われるまで、または文字の終わりまで１文字ずつ比較していきます。

戻り値：　比較の結果を次の値で返します。

負数　ｓ１がｓ２より小さい場合

0　ｓ１とｓ２が同じ場合

正数　ｓ１がｓ２より大きい場合

● strcpy

形式：　char＊　strcpy（char＊　ｓ１，const　char＊　ｓ２）；

機能：　文字列ｓ２を、文字列ｓ１にコピーします。コピーしたときオーバーフローチェックは行いません。

戻り値：　文字列ｓ１へのポインタを返します。

● strlen

形式：　int　strlen（const　char＊　string）；

機能：　文字列stringの長さを計算します。文字列の終わりの’￥0’は数えません。

戻り値：　stringの中の文字数を返します。

7．4　メモリ割り当て関数

● calloc

形式：　void＊　calloc（unsigned　n，unsigned　size）；

機能：　size バイトの大きさのメモリ領域をｎ個割り当てます。

戻り値：　割り当てられた領域のポインタを返します。割り当てに失敗した場合はＮＵＬＬを返します。

（注）確保サイズは、ＲＡＭサイズを超えないようにしてください。ＲＡＭサイズを超える値を指定した場合の戻り値は保証されません。

● malloc

形式：　void＊　malloc（unsigned　size）；

機能：　size バイトの大きさのメモリ領域を割り当てます。

戻り値：　割り当てられた領域のポインタを返します。割り当てに失敗した場合はＮＵＬＬを返します。

● free

形式：　void　free（void＊　ptr）；

機能：　calloc，malloc で確保されたメモリ領域を解放します。
ptr は、calloc，malloc により割り当てられたメモリ領域のポインタでなければなりません。

戻り値：　ありません。

7．5　数学関数

● abs

形式：　int　abs（int　x）；

機能：　整数の絶対値を求めます。

戻り値：　０～３２７６７の整数を返します。

● asin，acos，atan

形式：　double　asin（double　x）；
double　acos（double　x）；
double　atan（double　x）；

機能：　ｘに対する逆三角関数の値（角度）を計算します。
扱う角度の単位は angle 関数で（ＤＥＧ：度）、（ＲＡＤ）、（ＧＲＡＤ）を指定できます。
asin，acos では、引数ｘの値は－１～１の間でなければなりません。

戻り値：　次の表の範囲の値を返します。

逆三角関数	逆三角関数の値の範囲		
	ＤＥＧ	ＲＡＤ	ＧＲＡＤ
asin	−90°～ 90°	$-\pi/2 \sim \pi/2$	−100°～100°
acos	0°～180°	$0 \sim \pi$	0°～200°
atan	−90°～ 90°	$-\pi/2 \sim \pi/2$	−100°～100°

● asinh，acosh，atanh

形式：　double　asinh（double　x）；
double　acosh（double　x）；
double　atanh（double　x）；

機能：　逆双曲線関数の値を計算します。

戻り値：　計算の結果を返します。

● exp

形式：　double　exp（double　x）；

機能：　指数関数（e^x）を計算します。ただしｅは自然対数の底です。

戻り値：　e^x を返します。

● log，log１０

形式：　double　log（double　x）；
double　log１０（double　x）；

機能：　log　　ｘの自然対数を計算します。
log１０　ｘの常用対数を計算します。

戻り値：　計算結果を返します。

● pow

形式：　double　pow（double　x，double　y）；

機能：　ｘのｙ乗を計算します。

戻り値：　計算結果を返します。

● sin，cos，tan

形式：　double　sin（double　x）；
double　cos（double　x）；
double　tan（double　x）；

機能：　ｘに対する三角関数の値を求めます。
扱う角度の単位は angle 関数で（ＤＥＧ：度）、（ＲＡＤ）、（ＧＲＡＤ）を指定できます。

戻り値：　計算の結果を返します。

● sinh，cosh，tanh

形式：　double　sinh（double　x）；
double　cosh（double　x）；
double　tanh（double　x）；

機能：　双曲線関数の値を計算します。

戻り値：　計算の結果を返します。

● sqrt

形式：　double　sqrt（double　x）；

機能：　ｘの平方根を計算します。

戻り値：　計算の結果を返します。

7．6　ハードウェア・インタフェース関数

Ｉ／Ｏポート関数、機械語インタフェース関数は、誤って使うとプログラムやシステムエリアなどを破壊することがありますので、十分注意してください。

■ミニＩ／Ｏ関数

● miniget

形式：　int　miniget（void）；

機能：　ミニＩ／Ｏから１バイトのデータを読み込みます。
ビット２：Xin　ビット１：Din　ビット０：Ack

戻り値：　入力データを返します。

● miniput

形式：　void　miniput（char　byte）；

機能：　ミニＩ／Ｏに１バイトのデータを出力します。
ビット２：Busy　ビット１：Dout　ビット０：Xout

戻り値：　ありません。

■11ピンの８ビット制御関数

11ピンで８ビット制御を可能にします。

● fclose

形式：　int fclose（ＦＩＬＥ＊　stream）；

機能：　stream で指定されたファイルを閉じます。

戻り値：　正常に終了したときは０を返し、エラーになったときはＥＯＦを返します。

● fopen

形式：　ＦＩＬＥ　＊fopen（char　＊path, char　＊type）；

機能：　11ピンを８ビット制御モードに設定します。
path でデバイス名を、type でモードを指定します。
″pio″：８ビット制御　″r＋″、″w＋″または″a＋″：データの入出力

戻り値：　正常に終了したときはオープンしたストリームへのポインタを返します。
エラーになったときはヌルを返します。

● pioget

形式：　int pioget（void）；

機能：　11ピンから８ビットデータを入力します。

戻り値：　入力データを返します。

● pioput

形式：　void pioput（char byte）；

機能：　11ピンへ８ビットデータを出力します。

戻り値：　ありません。

● pioset

形式：　void pioset（char byte）；

機能：　８ビット制御の入出力モードを設定します。
１のときは入力モードの指定になり、０のときは出力モードの指定になります。

戻り値：　ありません。

■ＳＩＯ（ＲＳ-232Ｃ）制御関数

● fclose

形式：　int fclose（ＦＩＬＥ＊　stream）；

機能：　stream で指定されたファイルを閉じます。
全二重モードで使用しているときは、テキスト終了コード（ＴＥＸＴモードのＦＯＲＭＡＴで指定している終了コード）を書き込みます。

戻り値：　正常に終了したときは０を返し、エラーになったときはＥＯＦを返します。

● fopen

形式：　ＦＩＬＥ　＊fopen（char　＊path, char　＊type）；

機能：　11ピンをＲＳ-232Ｃ準拠で通信を行うモードに設定します。
（通信条件の設定は、ＴＥＸＴモードのＳＩＯで行ってください。）
デバイス名（path で指定）
″stdaux″：半二重通信の指定　″stdaux１″：全二重通信の指定
アクセスモード（type で指定）
″r＋″、″w＋″または″a＋″：データの入出力の指定

戻り値：　正常に終了したときはオープンしたストリームへのポインタを返します。
エラーになったときはヌルを返します。

(注) fopen () を使用する場合は、必ずfclose () で終了してください。fclose () を使用しない場合、送信データおよび終了コードが最後まで正しく送信されません。この場合、電源をオフにしたり、モードを切り替えても送信されません。

■通信バッファ制御

● feof

形式：　int feof（ＦＩＬＥ＊　stream）；

機能：　通信バッファのデータの有無を調べます。

戻り値：　データが最後に達していればー１を返し、そうでなければ０を返します。

■Ｉ／Ｏポート関数

● inport

形式：　unsigned　char　inport（unsigned　char　port）；

機能：　アドレス port のＩ／Ｏポートから１バイトのデータを読み込みます。
本機では０x２０から０x３ＦのＩ／Ｏポートが使用できます。

戻り値：　入力データを返します。

● outport

形式：　void　outport（unsigned　char　port, unsigned　char　byte）；

機能：　アドレス port のＩ／Ｏポートに１バイトのデータを出力します。
本機では０x２０から０x３ＦのＩ／Ｏポートが使用できます。

戻り値：　ありません。

■機械語インタフェース関数

● call

形式：　unsigned　call（unsigned　adr, void＊　arg__ＨＬ）；

機能： アドレス adr から始まる機械語を呼び出します。引数 arg＿ＨＬの値がＨＬレジスタに渡されます。

戻り値： ＨＬレジスタの値が返されます。

● peek

形式： unsigned char peek (unsigned adr)；

機能： アドレス adr から１バイトを読み込みます。

戻り値： 読み込んだデータを返します。

● poke

形式： void poke (unsigned adr, unsigned char byte)；

機能： アドレス adr に１バイトデータ byte を書き込みます。

戻り値： ありません。

７．７ データファイル関数

● fclose

形式： int fclose (ＦＩＬＥ＊ stream)；

機能： stream で指定されたファイルを閉じます。
″w″または″a″モードでオープンしているときは、終了コード（１Ａ）を書き込みます。

戻り値： 正常に終了したときは０を返し、エラーになったときはＥＯＦを返します。

● feof

形式： int feof (ＦＩＬＥ＊ stream)；

機能： ファイルの終わりを調べます。

戻り値： ファイルの終わりを検出したときは－１を返し、検出できなかったときは０を返します。

● flof

形式： unsigned long flof (ＦＩＬＥ＊ stream)；

機能： ファイルの未使用バイト数を求めます。

戻り値： 残りバイト数を返します。

● fopen

形式： ＦＩＬＥ ＊fopen (char ＊path, char ＊type)；

機能： データファイルにファイル番号を割り当て、オープンモードを指定し入出力を可能にします。
アクセスモード（type で指定）
″r″：データの読み出しの指定
″w″：データの書き込みの指定
″a″：データの追加書き込みの指定

戻り値： 正常に終了したときはオープンしたストリームへのポインタを返します。
エラーになったときはヌルを返します。

７．８ グラフィック関数

表示用のグラフィック関数です。機能はＢＡＳＩＣでの命令と同じです。引数などの詳細についてはＢＡＳＩＣの各命令の説明を参照してください。

● circle

形式： int circle (int x, int y, int r, double s-angle, double e-angle, double ratio, int reverse, unsigned short kind)；

機能： 円を描き、内部を塗りつぶします。開始角度から終了角度までを反時計方向に円を描きます。
開始角度、終了角度は10進度で指定します。
x, y： 円の中心点
r： 半径
s-angle： 開始角度
e-angle： 終了角度
ratio： 比率（Ｙ軸方向の半径／Ｘ軸方向の半径）
reverse： ドットの種類
　０：ドットをセット
　１：ドットをリセット
　２：ドットの反転
kind： 塗りつぶす模様の種類 ＢＡＳＩＣ命令と同じ
（第10章 ＢＡＳＩＣの各命令の説明を参照）

戻り値： 正常に終了したときは０を返し、エラーになったときは－１を返します。

（注）半径線を描くときは、開始角度や終了角度をマイナスで指定します。
0°の位置に半径線を描くときは、－360°と指定します。

● gcursor

形式： int gcursor (int x, int y)；

機能： グラフィックの表示開始位置をドット（点）単位で指定します。

戻り値： 正常に終了したときは０を返し、エラーになったときは－１を返します。

● gprint

形式： void gprint (char＊ image)；

機能： 指定されたドットパターンを表示します。

戻り値： ありません。

● line

形式： int line (int x_1, int y_1, int x_2, int y_2, int reverse, unsigned short mask, int rectangle)；

機能： 指定された２点間を線で結びます。

reverse：

０：ドットをセット
１：ドットをリセット
２：ドットの反転

mask：　ＢＡＳＩＣ命令と同じ
（第10章　ＢＡＳＩＣの各命令の説明を参照）

rectangle：

０：指定した mask で直線を描きます。
１：指定した mask で長方形を描きます。
２：指定した mask で長方形の内部を塗りつぶします。

戻り値： 正常に終了したときは０を返し、エラーになったときは－１を返します。

● paint

形式： int paint (int x, int y, unsigned short kind)；

機能： 囲まれた領域を塗りつぶします。

x, y：指定した点の座標

kind：塗りつぶす模様の種類　ＢＡＳＩＣ命令と同じ
（第10章　ＢＡＳＩＣの各命令の説明を参照）

戻り値： 正常に終了したときは０を返し、エラーになったときは－１を返します。

● point

形式： int point (int x, int y)；

機能： 指定したドットの状態を読み取ります。

戻り値： 指定したドットが点灯しているときは１を返し、消灯しているときは０を返します。指定したドットが画面外になるときは－１を返します。

● preset

形式： int preset (int x, int y)；

機能： 指定したドットを消します。

戻り値： 正常に終了したときは０を返し、エラーになったときは－１を返します。

● pset

形式： int pset (int x, int y, int reverse)；

機能： 指定したドットの点灯または反転を行います。

reverse：

０：ドットを点灯
１：ドットを反転

戻り値： 正常に終了したときは０を返し、エラーになったときは－１を返します。

７．９　その他の関数

● abort, exit

形式： void abort (void)；
void exit (int status)；

機能： 現在実行しているプログラムを終了します。

abort　プログラムを異常終了させます。画面に ＡＢＯＲＴ が表示されます。

exit　プログラムを正常終了させます。画面に ＥＸＩＴ に続いて status の値が表示されます。

戻り値： ありません。

● angle

形式： void angle (unsigned n)；

機能： n の値によって、三角関数の単位が設定されます。

n＝０　６０分法　：単位　ＤＥＧ（度）
n＝１　弧度法　：単位　ＲＡＤ（ラディアン）
n＝２　グラード法：単位　ＧＲＡＤ（グラード）

戻り値： ありません。

● breakpt

形式： void breakpt (void)；

機能： プログラムの実行を中断して、ブレークモードに入ります。

戻り値： ありません。

● clrscr

形式： void clrscr (void)；

機能： 画面をクリアし、カーソル（表示位置）を（０，０）にセットします。

戻り値： ありません。

● getch

形式： int getch (void)；

機能： １文字を stdin から直接読み込みます。読み込みに ↵ の入力は必要ありません。入力バッファが空のときは、次の入力があるまで待機します。読み込んだ文字は表示されません。

戻り値： 読み込んだ文字を返します。

● gotoxy

形式： void gotoxy (unsigned x, unsigned y)；

機能： 画面のカーソルを、座標（x，y）に移動します。
ただし、左上隅の座標を（０，０）とします。

戻り値： ありません。

● kbhit

形式： int kbhit (void)；

機能： キーボードからのキー入力の有無を調べます。

戻り値： キー入力が行われていれば０以外の値を返し、行われていなければ０を返します。

8．エラーメッセージ

8．1　コンパイルエラーメッセージ

エラーメッセージ	エ　ラ　ー　の　内　容
Null dimension	配列宣言で要素数が書かれていないものがある
array of function is illegal	要素が関数の配列を宣言している
can't find include file	インクルードファイルが見つからない
case not in switch	caseラベルがswitch文の中でない
constant expected	● 配列宣言で要素数が整数定数式でない ● caseラベルの式が整数定数式でない
default not in switch	defaultラベルがswitch文の中でない
define buffer full	＃defineが多すぎる
different s/u	構造体／共用体の代入で両辺の型が違う
division by 0	／または％演算子の右辺が０である
duplicate ＃define：名前	定義済みのマクロ名を定義している
duplicate case	switch文中に同じ値のcaseラベルがある
duplicate default	switch文中にdefaultラベルが複数ある
duplicate label：名前	定義済みのgotoラベルを定義している
empty character constant	文字定数構文中に文字が１つもない
float overflow	実数の定数値が大きすぎる
float underflow	実数の定数値が小さすぎる
function illegal in s/u	構造体／共用体の中で関数を宣言している
function returns illegal type	関数値として返せない型を定義している
if nest too deep	＃if／＃ifdef のネスティングが深すぎる
if nesting error	＃if／＃ifdef と＃endif の対応が取れていない
if -less elif	＃elif に対応する＃if／＃ifdef がない
if -less else	＃else に対応する＃if／＃ifdef がない
if -less endif	＃endif に対応する＃if／＃ifdef がない
illegal ＃ line	＃define行の構文が不正である
illegal break	break文がdo、for、while、switch文の中でない
illegal character	ソースプログラムに不正文字がある
illegal class	記憶クラスが不正である
illegal continue	continue文がdo、for、while文の中でない
illegal digit in octal	８進数内に不正数字（８と９）がある
illegal function	関数型でないもので関数呼び出しを行っている
illegal if	＃if／＃ifdef 行の構文が不正である
illegal include	＃include 行の構文が不正である
illegal indirection	単項＊演算子のオペランドが不正である

エラーメッセージ	エ　ラ　ー　の　内　容
illegal initialization	初期化の右辺が定数式でない
illegal main	main関数に引数を宣言している
illegal operand of 演算子	演算子のオペランドの型が不正である
illegal operand of U＋	単項＋演算子のオペランドの型が不正
illegal operand of U－	単項－演算子のオペランドの型が不正
illegal operand of ARG	関数呼び出しの引数の型が不正
illegal operand of RET	リターン文の式の型が不正
illegal s/u	構造体／共用体が不正使用されている
illegal size	構造体／共用体のサイズが大きすぎる
illegal switch expression	switch(e)の式のeの型が不正である
illegal type	不正な型変換が発生している
illegal void	void型の使用が正しくない
include nest too deep	＃include のネスティングが深すぎる
macro recursion	マクロが再帰している
memory full	メモリが不足している
missing argument：名前	関数定義の(　)内にない引数を宣言している
missing declarator	宣言子がない
missing function：名前	使用された関数が定義されていない
missing label	● goto文にラベルがない ● 使用されたgotoラベルが定義されていない
missing main	main(　)が定義されていない
missing member	● メンバー未定義の構造体／共用体を初期化している ● 式中で定義していないメンバーを使用している
missing member in s/u	構造体／共用体の定義でメンバーが１つもない
missing name in prototype	関数定義のプロトタイプに引数名がない
missing type	型を宣言していない
missing type in prototype	プロトタイプで構文規則違反がある
newline in character constant	文字定数の構文内に改行文字がある
newline in string constant	文字列定数の構文内に改行文字がある
prototype unmatch	関数呼び出しの式がプロトタイプに適合しない
redeclaration：名前	定義済みの名前を定義している
reserved：名前	この名前は予約されているので使用禁止
syntax error	構文規則に適合していない
token buffer full	マクロ展開が複雑すぎる
too complicated declarator	宣言が複雑すぎる
too complicated declaration	宣言が複雑すぎる
too complicated expression	式が複雑すぎる
too complicated initialize	初期化が複雑すぎる
too deep statement	文のネスティングが深すぎる

エラーメッセージ	エ　ラ　ー　の　内　容
too long initializer	初期化で文字列定数が長すぎる
too long macro	マクロ本文が長すぎる
too many ♯define	♯defineの個数が制限を超えている
too many case	caseラベルの個数が制限を超えている
too many characters in character constant	文字定数の文字数が制限を超えている
too many characters in string constant	文字列定数の文字数が制限を超えている
too many initializers	初期化式の個数が宣言より多い
too many label	gotoラベルの個数が制限を超えている
too many prototype	プロトタイプの個数が制限を超えている
unacceptable operand of &	&演算子のオペランドが不正である
undefine：名前	未定義の名前を使用している
unexpected EOF	構文規則の途中でソースプログラムが終了している
unknown size	サイズが確定できない
void function	void関数なのにreturn文で値を返している
zero or negative subscript	配列の要素数が0または負である

8．2　実行時エラーメッセージ

エラーメッセージ	エ　ラ　ー　の　内　容
NO MEMORY	関数の再帰などでメモリを使い果たした
BAD POINTER	ポインタでの代入、Ｃ領域以外のメモリへ書き込もうとした
DIVISION BY 0	0での割り算
UNKNOWN ERROR	ポインタでの代入で、Ｃ領域が破壊されている
BAD FUNCTION	ポインタでの関数呼び出しで、ポインタ値がおかしい
BAD STREAM	入出力ライブラリのストリームが適正でない
ARITHMETIC ERROR	● 数学ライブラリのシステムコールでエラーリターンした ● 浮動小数点数に関するシステムコールがエラーリターンした
FRAME ERROR	関数フレームが破壊された
I／O OPEN ERROR	ＳＩＯデバイスをオープンしているのに再度オープンした
I／O ERROR	ミニＩ／Ｏをオープンしていない

第７章　ＣＡＳＬ

１．ＣＡＳＬ（アセンブラ言語）

コンピュータ言語においてＢＡＳＩＣ、ＦＯＲＴＲＡＮ、ＣＯＢＯＬといった高水準な言語は、その記述がわかりやすいためプログラミングが容易であり、開発期間も短くてすみます。しかしこれらは、メモリの作業領域を多く必要としたり処理速度が遅いといった面があります。コンピュータの性能を最も効率よく活用できるのは機械語ですが、機械語コードで入力する必要があり、プログラミングは非常に手間がかかります。

アセンブラはこの機械語に最も近い言語です。ＢＡＳＩＣほど手軽ではないにしても、機械語に比べて使いやすく、コンピュータの性能を限界近くまで活用できます。ただ、アセンブラ言語で効率の良いプログラムを作るには、各命令を実行するときに、レジスタやフラグなどがどのような動きや働きをするのかを知る必要があります。これらの動き、働きなどはコンピュータのハードウエアに大きく左右され、メーカーや機種により異なるのが普通です。

このため経済産業省の情報処理技術者試験では、ＣＯＭＥＴという仮想の計算機を設定し、これのアセンブラ言語としてＣＡＳＬの仕様を取り決めたうえで試験問題を出しています。(平成13年度より、計算機はＣＯＭＥＴⅡに、アセンブラ言語はＣＡＳＬⅡに改定されていますが、本機はこれらに対応していません。)

本機のＣＡＳＬモードは、このＣＡＳＬの仕様に準拠したアセンブラ言語が扱えるため、情報処理技術者試験（基本情報技術者試験（旧第２種）では選択）のトレーニングとしてプログラミングからデバッグそしてシミュレーションまでの学習にお使いいただけます。

この取扱説明書は、本計算機の操作方法に重点をおいて説明しています。ＣＯＭＥＴⅡやＣＡＳＬⅡの仕様、ＣＡＳＬⅡの文法、プログラミングについてのくわしい説明は、有名書店で販売されている情報処理技術者試験のＣＡＳＬⅡに関する書籍を参照してください。

ＣＯＭＥＴの仕様については、265ページを参照してください。

２．ＣＡＳＬモードの構成

ＣＡＳＬモードは次の３つの機能から構成されています。(246ページの機能一覧表を参照ください。)

アセンブラ　：ＴＥＸＴエディタで作成したソースプログラムをオブジェクトプログラムに変換してメモリに書き込みます。
ＣＥ-126Ｐ（プリンタ）をお持ちの場合は、ＣＥ-126Ｐが接続され、“ＰＲＩＮＴ”が点灯しているとき、アセンブルした結果を印字できます。

モニタ　：レジスタの内容やオブジェクトを表示させたり、書き換えることができます。ブレークポイントのアドレスを設定することができます。
また、ＤＳ命令などで確保したメモリの内容を書き換えることができます。

シミュレーション　：オブジェクトプログラムを実行します。
ノーマル実行とトレース実行ができ、ＣＥ-126Ｐ（プリンタ）をお持ちの場合は、接続すれば実行結果を印字できます。
モニタ機能で指定したブレークポイントでの実行中断ができます。

2．1　プログラム作成手順

ＣＡＳＬのプログラミングは次の手順で行います。

【作成手順の解説】

①ＴＥＸＴエディタ機能を使用してソースプログラムを入力します。
　ＣＥ－126Ｐ（プリンタ）をお持ちの場合は、必要であればソースプログラムの印字を行います。本体のプログラムファイルエリアへの登録を行います。
②ＣＡＳＬモードのアセンブラ機能によりオブジェクトプログラムを生成します。
③モニタ機能を使用して、オブジェクトプログラムに必要なブレークポイントのアドレス設定やレジスタ、プログラムカウンタなどの設定を行います。
④シミュレータ機能でオブジェクトプログラムのノーマル実行、またはトレース実行を行います。
⑤再びモニタ機能を使用して、汎用レジスタやフラグレジスタの変化を確認します。また、オブジェクトやメモリ内容の確認と変更ができます。
⑥必要に応じて③～⑤を繰り返します。
⑦プログラムに誤りが発見されたときは①に戻り、ソースプログラムの変更を行い、再び②以降を実行します。
⑧必要に応じて①～⑦を繰り返します。

ＣＡＳＬモード機能一覧

この表のいずれかの状態（プログラム実行中を除く）でTEXTを押すことによって、テキストエディタ（ＴＥＸＴモード）に移ることができます。
なお、テキストエディタに移ったとき、生成されたオブジェクトプログラムは消去されます。

【参考】
ソースプログラムの作成
①TEXTを押してテキストモードにする。
②Eを押してソースプログラムを作成する。

SHIFT＋ASMBLを押してからCを押す

Assemble ― A → アセンブル実行
Monitor ― M → Register / Object
BREAK / CLS
Go ― G

Register ― R →

```
GR0：#XXXX  XXXX
GR1：#XXXX  XXXX
GR2：#XXXX  XXXX
GR3：#XXXX  XXXX
GR4：#XXXX  XXXX
PC ：#XXXX  XXXX
FR ：#XXXX  X
BP ：#XXXX  XXXX
BC ：#XXXX  XXXX
```

BREAK
▼▲で画面外のレジスタを呼び出します。
必要に応じて内容の書き換えができます。

Object ― O →

```
<<OBJECT>>
ADDRESS=#XXXX
```

#XXXX ↵ → オブジェクトプログラム表示
BREAK
▼▲でプログラムを送り、確認できます。
必要に応じて内容の書き換えができます。

Go ― G →

```
<<SIMULATION>>
START ADDRESS=#XXXX
```

#XXXX ↵ → Trace / Normal
T → トレース実行　↵でトレース実行を続けます。
N → ノーマル実行
ＥＸＩＴを実行した場合
BREAK

3．ソースプログラムの作成、編集（エディット）

ＣＡＳＬのソースプログラムの作成や変更などはＴＥＸＴ（テキスト）モードのエディタ機能を用いて行います。

[TEXT] [E] と押して、テキストエディタを選んでください。

```
TEXT EDITOR
<
```

● ＴＥＸＴモードのくわしい使いかたは、「第５章　ＴＥＸＴモード」を参照してください。

3．1　ソースプログラムの入力形式

ソースプログラムの構成を以下に示します。

```
32767BGN  ADD   0, DAT, 1;SAMPLE
```

行番号欄	ラベル欄（省略可）	命令コード欄	オペランド欄	注釈欄（省略可）
32767	BGN	ADD	0，DAT，1	；SAMPLE

(1)　行番号（ラインナンバー）欄

①行番号は１～65279までの数値を使用します。

②１～65279の範囲外の値を指定すると“ＬＩＮＥ　ＮＯ．ＥＲＲＯＲ”が表示されます。

(2)　ラベル欄

①ラベルは１～６文字まで使用できます。７文字以上を入力した場合は、先頭から６文字がラベルとして有効になります。また、79文字以上入力するとエラーになります。

②ＣＡＳＬでは、ラベルは６文字以内で先頭の文字はアルファベット大文字、２文字目以降はアルファベット大文字または数字と定義されています。

なお、ラベル欄にセミコロン（；）を書いて注釈行とすることもできます。

③ラベルは行番号に続けて入力する必要があります。

④ラベル入力後、[TAB] で次の入力位置へカーソルを移し、命令コードを入力します。

[TAB] について

ラベル、命令コード、オペランドの間には１文字以上のスペースが必要です。なお、ラベルを省略する場合は、行番号の後ろに１文字以上のスペースを入れる必要があります。このとき、[SPACE] の代わりに [TAB] を使うと、命令コードやオペランドの表示位置がそろって見やすくなります。

(3)　命令コード欄

①アルファベットキーで命令コードを入力します。入力後、[TAB] で次の入力位置へカーソルを移します。

(4)　オペランド欄

①オペランド欄はＧＲフィールド、adr フィールド、ＸＲフィールドで構成されます。

②各ブロック（フィールド）の区切りには、必ずコンマ（，）が必要です。

③ＸＲフィールドは省略することができます。

(5)　注釈欄

①プログラムに注釈を付加する場合は、オペランドの直後にセミコロン（；）を入力し、その後に注釈を入力します。

● 一行の長さは、注釈欄を含めて最大254文字までです。

3．2　ソースプログラムの消去

ＴＥＸＴモードの機能選択画面で、[D] を押せばデリート（Del）機能が選ばれ、次のようにテキスト内容を消去（削除）してよいかを聞いてきます。（テキスト内容がないときは、画面は変わりません。）

```
TEXT DELETE OK? (Y)
```

[Y] を押せば、テキスト内容がすべて消去され、ＴＥＸＴモードの機能選択画面に戻ります。

[Y] 以外のキー（そのとき有効に働くキー）を押せば、消去されずに機能選択画面に戻ります。

3．3　ソースプログラムの入力

ＴＥＸＴモードの機能選択画面で [E] を押せばテキストエディタが選ばれます。

```
TEXT EDITOR
<
```

このとき、[▼] または [▲] を押すと、ソースプログラムなどのテキスト内容がある場合は、画面に表示されます。何もないときは、画面は変わりません。

新たにソースプログラムを入力するときなどは、[ON]（BREAK）を押して機能選択画面に戻し、「3.2ソースプログラムの消去」の方法でテキスト内容を消去してください。

次にソースプログラムの入力手順を説明します。

①行番号を入力します。

②ラベルを入力するときは、行番号に続けて入力し [TAB] を押します。

ラベルがないときはそのまま [TAB] を押します。カーソルが命令コード欄に移ります。

③命令コードを入力します。その後 [TAB] を押せばカーソルがオペランド欄に移ります。

④オペランドは、コンマ（，）で区切って入力します。

⑤注釈は、オペランドの後にセミコロン（；）を入力して、その後に入力します。
ただし、DC命令の文字列中のセミコロンは注釈のための区切りとしては扱われません。（文字として扱われます。）

⑥1行の入力を完了したら [↵] を押して、プログラムをメモリに格納します。
次の行を入力するときは、①から繰り返します。

● すべての行の入力が終わったら [BREAK ON] で機能選択画面に戻します。

〈例〉次のソースプログラムを入力してみましょう。
「♠♥CARDS♦♣」を表示するプログラムです。

```
10L1     START  L2
20L2     OUT    DSP,N
30       EXIT
40N      DC     9
50DSP    DC     #E8
60       DC     #E9
70       DC     'CARDS'
80       DC     #EA
90       DC     #EB
100      END
```

[キー操作]	[解説]
[TEXT] [E]	テキストモードのエディタにする
10L1 [TAB] START [TAB] L2 [↵]	ラベル“L2”の行より実行開始
20L2 [TAB] OUT [TAB] DSP，N [↵]	DSP番地からN番地の文字数分を出力
30 [TAB] EXIT [↵]	プログラム実行の終了
40N [TAB] DC [TAB] 9 [↵]	出力する文字数「9」を格納
50DSP [TAB] DC [TAB] #E8 [↵]	「♠」マークのコードを16進数で格納
60 [TAB] DC [TAB] #E9 [↵]	「♥」マークのコードを16進数で格納
70 [TAB] DC [TAB] 'CARDS' [↵]	「CARDS」の文字列を格納
80 [TAB] DC [TAB] #EA [↵]	「♦」マークのコードを16進数で格納
90 [TAB] DC [TAB] #EB [↵]	「♣」マークのコードを16進数で格納
100 [TAB] END [↵]	プログラムの終わり（実行の終わりはEXIT）

以降の説明でこのプログラムを活用します。
入力完了後、[BREAK ON] で機能選択画面に戻ります。
なお、GR0～GR4はGRを省略して0～4のみを入力して指定することもできます。

4. アセンブル

TEXTモードで入力したソースプログラムを計算機が実行できるオブジェクトプログラムに変換する（アセンブルする）場合、まず、CASLモードにします。

[SHIFT] + [ASMBL] を押してから [C] を押します。
メニュー画面になります。

```
*** CASL ***
Assemble Monitor Go
```

[A] を押します。
アセンブルが実行されます。

```
*** CASL ***
Assemble Monitor Go
complete !
```

アセンブル実行中は画面の下の行に“assembling”と表示され、終われば“complete !”と表示されます。アセンブルするプログラムが短かいときは、この表示は一瞬で終わります。オブジェクトプログラムは1000番地から格納されます。

4.1 アセンブルリストをプリンタで印字する方法

CE-126P（プリンタ）をお持ちの場合は、本機に接続されて電源が入っているとき、メニュー画面で [SHIFT] + [P↔NP] と押すと画面右下に“PRINT”が点灯します。再度、[SHIFT] + [P↔NP] と押すと消えます。“PRINT”が点灯しているときにアセンブルを実行すると、アセンブルリストが印字されます。

〈印字例〉

```
ADD : OBJECT   : LINE NO.
               : 10
1000:7000 100B: 20
1002:7000 100A: 20
1004:8000 0002: 20
1006:1244 0002: 20
1008:6400 0004: 30
100A:0009      : 40
100B:00E8      : 50
100C:00E9      : 60
100D:0043      : 70
100E:0041      : 70
100F:0052      : 70
1010:0044      : 70
1011:0053      : 70
1012:00EA      : 80
1013:00EB      : 90
               : 100
|←①→|←──②──→|←──③──→|
LABEL :ADDRESS ↑
L1     1000    |
L2     1000    ④
N      100A    |
DSP    100B    ↓
```

①アドレス（16進）
②オブジェクト（16進）
③対応するソースプログラムの行番号
④ラベルテーブル

アセンブルリストにはソースプログラムは印字されません。

4．2 エラーメッセージ

アセンブルの実行で、ソースプログラムにエラーが検出されると、下表のようなエラーメッセージが表示されます。

エラーは CLS で解除して、TEXTエディタでソースプログラムを修正してください。

アセンブル時に使用するメモリは、オブジェクトを記憶するメモリを除いて、ラベル1つで8バイト、ワーク用として4バイト、スタック用として64バイトです。メモリが足りないとメモリエラーになります。

エラーの種類	エラーメッセージ	原　　因
オペコードエラー	OP－CODE　ERROR（行番号）	行番号で示すプログラムの命令コードに誤りがある
	OP－CODE　ERROR（0）	ソースプログラムがない
オペランドエラー	OPERAND　ERROR（行番号）	行番号で示すプログラムのオペランドに誤りがある
ラベルエラー	LABEL　ERROR（行番号）	行番号で示すプログラムのラベルに誤りがある
メモリエラー	MEMORY　ERROR（0）	● オブジェクトを記憶するメモリが不足している ● スタック用メモリが不足している
その他のエラー	OTHER　ERROR	プログラムの先頭にSTART命令がない、終わりにEND命令がないなど、入力形式が正しくない

5．シミュレーション

メニュー画面で G を押すと、オブジェクトプログラムの実行処理に移ります。

G

```
<< SIMULATION >>

START ADDRESS=#1000
```

実行開始アドレス

実行開始アドレスは、START命令で指定されたアドレスが表示されます。START命令で実行開始アドレスが指定されていないときは、自動的に＃1000が実行開始アドレスになります。

表示されている実行開始アドレスを変更する場合は、上記のアドレス表示状態で、変更したいアドレスを10進数、16進数、またはラベルで入力します。

次に ↵ を押せば、実行方法選択画面になります。

↵

N または T で選択します。

5．1 ノーマル実行

実行方法選択画面で N を押せば、設定された実行開始アドレスからプログラムの実行を開始します。

N

```
♠♥CARDS♦♣
```

↵

```
*** CASL ***

Assemble Monitor Go
```

EXIT命令が実行されると、メニュー画面に戻ります。

ノーマル実行中、次の場合はプログラムの実行を停止して、実行内容の表示や停止したアドレスを表示します。

- OUT命令を実行した場合。↵ を押せばプログラムの継続実行ができます。
- ブレークポイント（BP）が設定されていて、プログラムカウンタ（PC）の値がブレークポイントの値と一致した場合。この場合、ブレークポイントが設定されているアドレスの命令は実行されていません。
- BREAK が押された場合。この場合、プログラムカウンタに停止（ブレーク）したときのアドレスが入っているので、↵ を押せばプログラムの継続実行ができます。また、CLS や BREAK でメニュー画面に戻してから、再び実行操作をしても継続実行できます。
- ＊命令を実行した場合（263ページ参照）
  ＊命令により停止し、レジスタ内容などが表示されているときに ↵ を押すと、＊命令をスキップして次の命令を実行します。
  ＊命令（第1語目が＃00XX、第2語目が＃XXXXの命令語）を実行した場合、プログラムカウンタには第1語目のアドレスが入っています。したがって BREAK でプログラムを中断した後、オブジェクトの修正、または実行アドレスの修正を行わずにプログラムを再実行させると、再度同じアドレスでプログラムが停止します。このような場合は、命令語または実行アドレスを修正した後、プログラムを実行してください。
- 画面右下に“PRINT”が点灯しているときは、OUT命令による結果が印字されます。

5．2 トレース実行

実行方法選択画面で T を押せば、表示されている実行開始アドレスからプログラムのトレース実行が開始されます。トレース実行中は1命令実行ごとに、実行結果（GR0～GR4、PC、FRの内容）を表

示して停止します。ここで表示されるPCの値は命令実行後のPCの値、つまり次に実行する番地です。次に[↵]を押すと、次の命令を実行して停止します。

```
1000:  GR0:0000  GR4:1B0B
       GR1:0000  PC :1002
       GR2:0000  FR :0000
       GR3:0000  <PUSH>
```

（注）GR4の値はメモリの使用状態により変わります。

[↵]

```
1002:  GR0:0000  GR4:1B0A
       GR1:0000  PC :1004
       GR2:0000  FR :0000
       GR3:0000  <PUSH>
```

実行した命令

メニュー画面で[SHIFT]＋[P↔NP]を押し、“PRINT”が点灯しているときは、以下のフォーマットで実行結果が（16進数で）印字されます。

〈印字例〉

```
ADD :GR0  GR1  GR2  GR3
1000:0000 0000 0000 0000
1002:0000 0000 0000 0000
1004:0000 0000 0000 0000
♠♥CARDS♦♣
0002:0000 0000 0000 0000
1006:0000 0000 0000 0000
1008:0000 0000 0000 0000
←①→|←②→|←③→|←④→|←⑤→|
```

①アドレス
②GR0
③GR1
④GR2
⑤GR3

5．3 シミュレーションでのエラー

シミュレーションでは、次のエラーが発生することがあります。

エラーメッセージ	原因
OBJECT ERROR	オブジェクトプログラムがない。
＊MEM＊ ＊ERR＊	● JMP命令などで使用可能な領域を超えたアドレスへのジャンプが行われた。 ● 使用可能な領域を超えて、データのロードやセーブをしようとした。
＊OPR＊ ＊ERR＊	OUT命令で出力文字を97文字以上にした。

6．モニタ

モニタ機能では、仮想計算機COMETの各レジスタの内容確認や、オブジェクトプログラムを表示できます。

また、レジスタの値やオブジェクトの設定・変更、およびシミュレーションのためのブレークポイントの設定などができます。

モニタ機能は、メニュー画面で[M]を押すことによって選べます。（[SHIFT]＋[ASMBL]を押してから[C]を押すとメニュー画面になります。）

[M]

```
<< MONITOR >>

Register    Object
```

モニタ画面では、次の操作が可能です。

キー操作	機能
[R]	レジスタの内容の表示と、レジスタへの値の設定ができます。
[O]	オブジェクトプログラムと逆アセンブルした命令の表示がでます。また、この内容を設定したり、変更したりできます。

6．1 レジスタの内容の表示

モニタ画面で[R]を押すとレジスタの内容やブレークポイントの内容が表示されます。

[R]

```
GR0  #0000  0
GR1 :#0000  0
GR2 :#0000  0
GR3 :#0000  0
GR4 :#1B0B  6923
PC  :#1000  4096
```

レジスタ　16進数で内容を表示　10進数で内容を表示

（注）GR4の値はメモリの使用状態により変わります。

以降、[▼]でカーソル（このときのカーソルはコロン（：）の消灯）を下に移動していけば、FR以降を画面に呼び出すことができます。

⋮

[▼]

```
GR3 :#0000  0
GR4 :#1B0B  6923
PC  :#1000  4096
FR  :#0000  0
BP  :#FFFF  65535
BC   #0000  0
```

戻すときは[▲]を押します。[▼]の代わりに[↵]を押しても同様に働きます。

各レジスタ名は以下のとおりです。

表　示	レジスタ名	機　能
GR0～GR4	汎用レジスタ	算術演算と論理演算に用います。GR1～GR4は指標レジスタとしても用います。また、GR4はスタックポインタとしても用います。
PC	プログラムカウンタ	次に実行すべき命令が格納されているアドレスを記憶します。
FR	フラグレジスタ	特定の命令を実行した結果が、正、ゼロ、負になると、それぞれ0、1、2にセットされます。
BP	ブレークポインタ	シミュレーションでの実行を制御するために使用します。
BC	ブレークカウンタ	

6．2　レジスタに値を設定する方法

GR0～GR4、PC、FR、BP、BCの各レジスタに値を設定できます。
値を設定するときは、レジスタにカーソル（コロン（：）の消灯）を移し、そのまま、10進数、16進数、ラベル、文字などで入力して⏎を押します。

〈例〉10進数の入力：　１２３⏎　　＃００７Ｂ　１２３　が入力されます。
　　16進数の入力：＃００７Ｂ⏎　　＃００７Ｂ　１２３　が入力されます。
　　ラベルの入力：　"Ｌ１"　⏎　　ラベル"Ｌ１"が定義されているアドレスが入力されます。
　　文字の入力　：　'Ａ'　⏎　　Ａの文字コード　＃００４１　６５　が入力されます。

設定終了後、BREAKでモニタ画面に戻ります。

レジスタに値を設定するときは、次の点に注意してください。

①設定する値は10進数、16進数、ラベル、文字で入力します。
②負数は先頭に⊖でマイナスを入力します。
③設定できる値の範囲は−32768～65535までの整数です。この範囲を超えた値を入力して⏎を押すと、入力前の値に戻ります。
④FRの値は0、1、2だけです。0、1、2以外の値が入力されると、他のビットは無視されます。ただし、画面には入力した値が表示されます。
⑤アセンブル実行後の各レジスタには、以下の値が設定されています。

GR0～GR3：0
GR4　　：オブジェクトエリアの最上位番地＋1
PC　　：メインプログラムの実行開始アドレス
　　　　（STARTで指定されたラベルのアドレス）
FR　　：0
BP　　：FFFF（16進）　65535（10進）
　　　　（ブレークポイントが設定されていない状態）
BC　　：0

（注）GR4はスタックポインタとして最大限に使えるように、オブジェクトエリアの最上位番地＋1に指定されています。この値はモニタ機能で呼び出してマニュアル操作で変更、またはプログラム上で変更できます。

6．3　オブジェクトコードの表示

アセンブルの実行によって作成されたオブジェクトプログラムを表示させることができます。メニュー画面でMを押すとモニタ画面になり、モニタ画面でOを押すとオブジェクトプログラムが記憶されているときは、アドレス＃１０００が表示されます。

O

```
<< OBJECT >>

ADDRESS=#1000
```
メインプログラムの開始アドレス

次に⏎を押すと、オブジェクトエリアの＃１０００番地に記憶された内容から表示されます。表示内容は、オブジェクトコードと逆アセンブルしたニモニックです。

⏎
▼
⋮
▼

▼でカーソル（コロン（：）の消灯）を移動していけば、以降のアドレスの内容を見ることができます。
また、▲で戻すことができます。
なお、▼の代わりに⏎を使用しても同じ働きになります。

オブジェクトプログラムが記憶されていないときに、モニタ画面でOを押した場合はエラーになり"OBJECT ERROR"が表示されます。
このときはCLSを押してメニュー画面に戻し、アセンブルを行ってください。

（注）オブジェクトプログラムは、CASLモードから他のモード（TEXT、RUN）へ切り替わったときに消去されます。
他のモードからCASLモードに切り替えたときは、再度アセンブルを行ってください。

OUT命令について

本機のマクロ命令のOUTa，bは、PUSHa、PUSHb、CALL2とLEA4，2，4の命令群を生成します。
くわしくは271ページを参照してください。

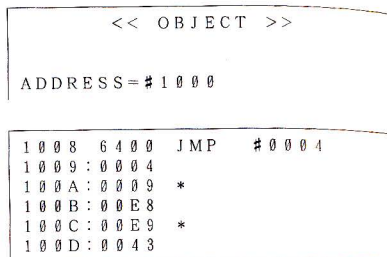
６．４　任意のアドレス内容の表示

メインプログラムの開始アドレスを表示している画面で、表示させたいアドレスを10進数、16進数、またはラベルで入力し、[↵]を押すと入力したアドレスの内容が表示されます。

```
<< OBJECT >>

ADDRESS=#1000
```

＃１００８[↵]

```
1008 6400 JMP  #0004
1009:0004
100A:0009 *
100B:00E8
100C:00E9 *
100D:0043
```

[BREAK]を押せば、モニタ画面に戻ります。

（注）● 入力したアドレスを命令語の第１語目とみなしますので、逆アセンブルの内容は正しく表示されない場合があります。

● 逆アセンブルした結果、ニモニックで表示する内容がない場合は ＊ マークを表示します。
オブジェクトプログラムがないアドレスは、００００　＊表示になります。

● 入力できるアドレスは＃１０００から、アセンブル終了時にＧＲ４に入っているアドレス－１の範囲です。

６．５　オブジェクトコードの書き換え

カーソル（コロン（：）の消灯）を書き換えたいアドレスへ移し、10進数、16進数、ラベル、文字でオブジェクトコード、またはそれに相当する内容を入力することにより、オブジェクトコードを書き換えることができます。

〈例〉書き換えたいアドレスの行を表示させ（：の消灯）

＃１２１０[↵]

のように操作します。

オブジェクトコードが書き換えられ、ニモニックも変わります。

また、アセンブラの命令語も入力できます。

● 10進数、16進数、ラベル、文字での入力の形式は255ページの「レジスタに値を設定する方法」を参照してください。

７．ＣＡＳＬ実行例

次のプログラムは５個のデータの総和を求めるプログラムです。

データは、行番号130～170に定義されています。また、行番号120で総和を格納する領域を確保し、行番号90でその領域に総和を収納しています。


```
10EXAM  START
20BGN   LEA    GR0, 0        (GR0をクリアする)
30      LEA    GR1, 0        (GR1をクリアする)
40      JMP    AGN1          (無条件にAGN1へジャンプする)
50AGN   ADD    GR0, DAT, GR1 (指標レジスタGR1を利用してデータを順番に
                              GR0へ加算する)
60      LEA    GR1, 1, GR1   (GR1の内容に1を加えて、結果をGR1に入
                              れる　カウントアップ)
70AGN1  CPA    GR1, N        (GR1の内容とN番地の内容とを大小比較する
                              個数の判断)
80      JMI    AGN           (FRの値が“負”を示していればAGNへジャ
                              ンプする)
90      ST     GR0, TTL      (GR0の内容をTTL番地へ格納する)
100     EXIT                 (プログラムの実行終了)
110N    DC     5
120TTL  DS     1             (結果を格納するためのエリアを確保する)
130DAT  DC     #000C  ┐
140     DC     #07F3  │
150     DC     #0231  ├ サンプルデータ
160     DC     #0009  │
170     DC     #000F  ┘
180     END                  (プログラム終了)
```

このプログラムをＴＥＸＴモードで入力し、ＣＡＳＬモードでアセンブルしてください。

７．１　オブジェクトコードの内容確認

以下の手順に従って、モニタ機能によりオブジェクトコードの内容を確認します。

キー操作	表　示
	＊＊＊ ＣＡＳＬ ＊＊＊ Assemble Monitor Go
[M]	＜＜ ＭＯＮＩＴＯＲ ＞＞ Register Object
[O]	＜＜ ＯＢＪＥＣＴ ＞＞ ＡＤＤＲＥＳＳ＝＃１０００
[↵]	1000 1200 LEA 0, #0000 1001:0000 1002:1210 LEA 1, #0000 1003:0000 1004:6400 JMP #100A 1005:100A

▼ ⋮	1006:2001 ADD 0, #1014, 1 1007:1014 1008:1211 LEA 1, #0001, 1 1009:0001 100A:4010 CPA 1, #1012 100B:1012 100C:6100 JMI #1006 100D:1006 100E:1100 ST 0, #1013 100F:1013 1010:6400 JMP #0004 1011:0004 1012:0005 * 1013:0000 1014:000C * 1015:07F3 1016:0231 ***** 1017:0009 1018:000F * 1019:0000
BREAK ON	<< MONITOR >> Register Object

7．2 ノーマル実行

次にソースプログラムの行番号８０ JMI AGN にブレークポイントを設定して、プログラムの実行結果を調べることにします。
前項の方法でオブジェクトコードを調べると、この命令はアドレス＃１００Ｃに格納されていることがわかります。

キー操作	表示	説明
	*** CASL *** Assemble Monitor Go	
M	<< MONITOR >> Register Object	モニタ機能を選びます。
R ▼ ⋮ ▼	GR0 #0000 0 GR1:#0000 0 GR2:#0000 0 GR3:#0000 0 GR4:#1AA5 6821 PC :#1000 4096 BP #FFFF 65535	レジスタの内容を呼び出し、カーソル（コロン（：）の消灯）をブレークポイントレジスタへ移します。 ブレークポイントは設定されていません。
#100C	BP #FFFF #100C_	ブレークポイントをアドレス＃１００Ｃに設定します。
↵	BP #100C 4108	

▼	BC #0000 0	ブレークカウンタを２に設定します。（２回目の実行で停止）
2 ↵	BC #0002 2	
BREAK BREAK	*** CASL *** Assemble Monitor Go	メニュー画面に戻します。
G ↵	<< SIMULATION >> START ADDRESS=#1000 Normal Trace	実行方法選択画面にします。
N	100C: GR0:000C GR4:1AA5 * * GR1:0001 PC :100C *STP* GR2:0000 FR :0002 * * GR3:0000 <JMI>	ノーマル実行をします。２回目実行の＃１００Ｃ番地でブレークがかかり、レジスタの内容を表示します。ＧＲ０にデータ＃０００Ｃが加えられます。
↵	100C: GR0:07FF GR4:1AA5 * * GR1:0002 PC :100C *STP* GR2:0000 FR :0002 * * GR3:0000 <JMI>	実行が再開され、次のデータ（＃０７Ｆ３）をＧＲ０に加え、ＧＲ１は２になります。
BREAK BREAK	*** CASL *** Assemble Monitor Go	
M R	GR0 #07FF 2047 GR1:#0002 2 GR2:#0000 0 GR3:#0000 0 GR4:#1AA5 6821 PC :#100C 4108	モニタでレジスタを呼び出します。
▼ ⋮ ▼	BC #0000 0	カーソルをブレークカウンタへ移します。
4 ↵	BC #0004 4	４を設定します。
BREAK BREAK	*** CASL *** Assemble Monitor Go	メニュー画面に戻します。
G ↵	<< SIMULATION >> START ADDRESS=#100C Normal Trace	実行方法選択画面にします。実行開始アドレスは先ほど、停止したアドレスになっています。
N	100C: GR0:0A48 GR4:1AA5 * * GR1:0005 PC :100C *STP* GR2:0000 FR :0001 * * GR3:0000 <JMI>	ノーマル実行をします。先に停止した状態からの継続実行になります。ＧＲ０には５個のデータの合計が入っています。
BREAK BREAK	*** CASL *** Assemble Monitor Go	メニュー画面に戻します。

(注) ● プログラムカウンタ（PC）とブレークポイント（BP）の値が一致したときに、ブレークカウンタ（BC）の値が0または1であればプログラムの実行を停止し、ブレークカウンタの値を0にします。ブレークカウンタの値が0または1以外のときは、ブレークカウンタから1を減じて、プログラムを続行します。

BC＝2のときの実行状況

実行行番号	20行 → 30行 → 40行 → 70行 → 80行 → 50行 → 60行 → 70行 → 80行 →
BC＝2	2→1　　　　　　1→0
GR0＝0	0→000C　　　　　000C

（↑ 停止）

● スタートアドレスとブレークポイント（BP）の値が等しいとき、ブレークカウンタ（BC）の値が0であればプログラムを続行し、次にプログラムカウンタとブレークカウンタの値が一致したときに停止します。
また、ブレークカウンタの値が1であれば、何も実行せずに停止し、ブレークカウンタの値を0にします。ブレークカウンタの値が0および1以外のときは、1を減算してプログラムを続行します。

7．3　ブレークポイントの解除

設定したブレークポイントの解除には、次の2通りの方法があります。

①アセンブルを再実行します。

②ブレークポイントの設定時と同じように、メニュー画面より
[M][R][▼]…と操作して、BPレジスタを表示させます。次に－1 [↵] と操作すると解除できます。
[BREAK] でメニュー画面に戻します。

7．4　トレース実行

同じプログラムをトレース実行してみましょう。
ブレークポイントは解除してください。なお次の例はアセンブル実行直後の例を示しています。

キー操作	表　　示	説　　明
	＊＊＊ CASL ＊＊＊ Assemble Monitor Go	
[G][↵]	<< SIMULATION >> START ADDRESS=#1000 Normal　Trace	
[T]	1000:GR0:0000 GR4:1AA5 GR1:0000 PC :1002 GR2:0000 FR :0001 GR3:0000 <LEA>	トレースモードで実行を開始します。 LEA 0, 0でGR0をリセットします。
[↵]	1002:GR0:0000 GR4:1AA5 GR1:0000 PC :1004 GR2:0000 FR :0001 GR3:0000 <LEA>	LEA 1, 0でGR1をリセットします。

キー操作	表　　示	説　　明
[↵]	1004:GR0:0000 GR4:1AA5 GR1:0000 PC :100A GR2:0000 FR :0001 GR3:0000 <JMP>	JMP AGN1に無条件に分岐（ジャンプ）します。PCの値を見れば100A番地であることがわかります。
[↵]	100A:GR0:0000 GR4:1AA5 GR1:0000 PC :100C GR2:0000 FR :0002 GR3:0000 <CPA>	CPA 1, NでGR1の内容とN番地の内容を比較します。
[↵]	100C:GR0:0000 GR4:1AA5 GR1:0000 PC :1006 GR2:0000 FR :0002 GR3:0000 <JMI>	JMI AGNでFRが2のときはAGNアドレスへ分岐します。
[↵]	1006:GR0:000C GR4:1AA5 GR1:0000 PC :1008 GR2:0000 FR :0000 GR3:0000 <ADD>	ADD 0, DAT, 1で指標レジスタ（GR1）を利用してデータをGR0に加算します。
[↵]	1008:GR0:000C GR4:1AA5 GR1:0001 PC :100A GR2:0000 FR :0000 GR3:0000 <LEA>	GR1に1を加算します。
[↵] ⋮ [↵]	100A:GR0:000C GR4:1AA5 GR1:0001 PC :100C GR2:0000 FR :0002 GR3:0000 <CPA> 100A:GR0:0A48 GR4:1AA5 GR1:0005 PC :100C GR2:0000 FR :0001 GR3:0000 <CPA>	以降、同様な動作がデータ数分繰り返されます。
[↵]	100C:GR0:0A48 GR4:1AA5 GR1:0005 PC :100E GR2:0000 FR :0001 GR3:0000 <JMI>	CPA 1, NでFRが1となったため、分岐せずに次の命令を実行します。
[↵]	100E:GR0:0A48 GR4:1AA5 GR1:0005 PC :1010 GR2:0000 FR :0001 GR3:0000 <ST>	ST 0, TTLでGR0の値を番地TTLへ格納します。
[↵]	1010:GR0:0A48 GR4:1AA5 GR1:0005 PC :0004 GR2:0000 FR :0001 GR3:0000 <JMP>	EXIT命令により実行を停止します。（EXIT処理ルーチン#0004番地へ分岐）
[↵]	＊＊＊ CASL ＊＊＊ Assemble Monitor Go	メニュー画面に戻ります。

7．5　空欄穴埋め問題の実行例

前の例題でプログラムの行番号６０が空欄穴埋め問題となっていた場合、下の例のように６０ラインに＊を入力します。＊命令は［＊］を押して入力します。

〈例〉

```
10EXAM   START
20BGN    LEA    GR0,0
30       LEA    GR1,0
40       JMP    AGN1
50AGN    ADD    GR0,DAT,GR
  1
60       LEA    GR1,1,GR1
70AGN1   CPA    GR1,N
80       JMI    AGN
90       ST     GR0,TTL
100      EXIT
110N     DC     5
120TTL   DS     1
130DAT   DC     #000C
140      DC     #07F3
150      DC     #0231
160      DC     #0009
170      DC     #000F
180      END
```

⇨

〈例〉

```
10EXAM   START
20BGN    LEA    GR0,0
30       LEA    GR1,0
40       JMP    AGN1
50AGN    ADD    GR0,DAT,GR
  1
60       *
70AGN1   CPA    GR1,N
80       JMI    AGN
90       ST     GR0,TTL
100      EXIT
110N     DC     5
120TTL   DS     1
130DAT   DC     #000C
140      DC     #07F3
150      DC     #0231
160      DC     #0009
170      DC     #000F
180      END
```

上記、右のプログラムをアセンブルすると行番号６０に対応するオブジェクトコードが“＃００００　＃００００”になります。

〈例〉

```
ADD : OBJECT   : LINE NO.
               : 10
1000:1200 0000: 20
1002:1210 0000: 30
1004:6400 100A: 40
1006:2001 1014: 50
1008:0000 0000: 60
100A:4010 1012: 70
100C:6100 1006: 80
100E:1100 1013: 90
1010:6400 0004: 100
1012:0005      : 110
1013:0000      : 120
1014:000C      : 130
1015:07F3      : 140
1016:0231      : 150
1017:0009      : 160
1018:000F      : 170
               : 180
```

このプログラムを実行すると、＃１００８番地でプログラムの実行を停止しますので、レジスタの値を確認できます。また、モニタ機能を使って、オブジェクトの確認や修正ができます。

キー操作	表示	説明
	＊＊＊　CASL　＊＊＊ Assemble　Monitor　Go	
［G］［↵］	<<　SIMULATION　>> START　ADDRESS＝＃１０００ Normal　　Trace	
［N］	１００８：　GR０：０００C　GR４：１AAC ＊　　＊　GR１：００００　PC　：１００８ ＊STP＊　GR２：００００　FR　：００００ ＊　　＊　GR３：００００　<＊>	＊命令があるとプログラムの実行を停止します。
［↵］	１００８：　GR０：００１８　GR４：１AAC ＊　　＊　GR１：００００　PC　：１００８ ＊STP＊　GR２：００００　FR　：００００ ＊　　＊　GR３：００００　<＊>	（注１） （注２）
［BREAK］［BREAK］	＊＊＊　CASL　＊＊＊ Assemble　Monitor　Go	
［M］［O］ ＃１００８	<<　OBJECT　>> ADDRESS＝＃１００８_	アドレス＃１００８のオブジェクトプログラムを呼び出します。
［↵］	１００８　００００　＊ １００９：００００ １００A：４０１０　CPA　　１，＃１０１２ １００B：１０１２ １００C：６１００　JMI　　＃１００６ １００D：１００６	
LEA［TAB］GR１，１，GR１	１００８　００００　LEA　GR１，１，GR１_ １００９：００００ １００A：４０１０　CPA　　１，＃１０１２ １００B：１０１２ １００C：６１００　JMI　　＃１００６ １００D：１００６	空欄の答えを入力します。
［↵］	１００８　１２１１　LEA　　１，＃０００１，１ １００９：０００１ １００A：４０１０　CPA　　１，＃１０１２ １００B：１０１２ １００C：６１００　JMI　　＃１００６ １００D：１００６	アセンブルし、オブジェクトが生成されます。
［BREAK］［BREAK］［G］ "BGN"	<<　SIMULATION　>> START　ADDRESS＝"BGN"_	もう一度初めからプログラムを実行します。
［↵］	<<　SIMULATION　>> START　ADDRESS＝＃１０００ Normal　　Trace	
［N］	＊＊＊　CASL　＊＊＊ Assemble　Monitor　Go	終了するとメニュー画面に戻ります。
［M］［O］"TTL" ［↵］	１０１３　０A４８　＊＊＊＊＊ １０１４：０００C １０１５：０７F３　＊＊＊＊＊ １０１６：０２３１ １０１７：０００９　＊ １０１８：０００F	（注３）

（注1）シミュレーションで＊命令を実行し、プログラムが停止しているときに[↵]を押すと、次の命令から実行を開始します。実行後、＊命令があると再び停止します。
（注2）この時点でGR0には＃07FF、GR1には＃0001が入っているべきですが、値が異なっていることがわかります。
（注3）合計を確認すると正しい値が入っています。つまり、空欄に入れた答えが正しいことがわかります。この時点では、オブジェクトコードは変更されていますが、ソースプログラムは変更されていません。ソースプログラムはテキストモードに戻って変更します。

8．COMETの仕様

8．1　仮想計算機COMETと本機CASLとの相異点

本機のCASLの仕様は、経済産業省の設定している仮想計算機COMETのCASLと比べて、以下の点が異なります。（平成13年度より、COMETはCOMETⅡに、CASLはCASLⅡに仕様改定されています。）

(1)START命令

ラベルが省略できます。また、オペランドが省略された場合、＃1000番地からプログラムを実行します。

(2)DC命令

オペランドが10進定数で−32768〜65535の範囲にないときは、アセンブルするとエラーになります。

(3)IN命令（CALL　＃0000）

IN命令を実行すると、画面に“？”が表示され、キーからの入力が可能になります。このとき、[▼]の入力があると、入力文字長に−1（＃FFFF）を設定します。
この機能は、入力装置としてカードリーダを想定したときに利用すると便利です。

(4)OUT命令（CALL　＃0002）

OUT命令を実行すると、“PRINT”が点灯しているときは印字し、点灯していないときは画面に表示します。
出力文字数が97文字以上のときはエラーになります。

(5)WRITE命令（CALL　＃0006）

WRITE命令を実行するとレジスタの内容を表示し、プログラムの実行を停止します。このとき[↵]を押せばプログラムの実行を続行します。

(6)複数プログラムの連結

本機には、別々にアセンブルして作成したオブジェクトプログラムを連結する“リンカ”の機能はありません。このような動作をさせたいときは、個々にSTART命令とEND命令で囲んだ複数のプログラムを一度にアセンブルしてください。なお、アセンブルするとき、他のプログラムのラベルと同じものがあるとエラーになります。

8．2　COMETの仕様概略

アセンブラ言語の理解を深めるため、本機のCASLモードでの仮想ハードウェア仕様（COMETに準拠）について概略を説明します。

(1)1語長　：16ビット
(2)制御方式　：逐次制御方式（ノイマン型）
(3)数値の表現：16ビットの2進数、負数は2の補数表示
(4)レジスタ　：①GR0〜4（16ビット）　General Register　汎用レジスタ
　GR1〜GR4は指標レジスタとしても使用。さらにGR4はスタックポインタとしても使用される。
②PC（16ビット）　Program Counter　プログラムカウンタ
　実行中の命令語の先頭アドレスを保持するレジスタ。
③FR（2ビット）　Flag Register　フラグレジスタ
　演算、比較結果の情報保持レジスタ。

	GRに設定されたデータ		
	正	ゼロ	負
FRの値	00	01	10

（2進表記）

［補足］トレース実行を行うとレジスタの内容を容易に確認できます。

なぜ01がゼロなのか

LEA命令でGRに設定される値がゼロのとき、FRには01が設定されます。
そのため、FRの01をゼロと呼び、JZE（Jump on ZEro）で分岐します。
分岐命令においては注意してください。

8．3　命令語の構成

CASLの命令は2語長と定義されています。その構成をまとめます。

第1語				第2語	命令語とアセンブラとの対応		
OP		GR	XR	adr	書きかた		ニモニックスペル
主	副				命令コード	オペランド	
0	0				未使用		
1	0				LD	GR，adr，XR	load
1	1				ST	GR，adr，XR	store
1	2				LEA	GR，adr，XR	load effective address
2	0				ADD	GR，adr，XR	add arithmetic
2	1				SUB	GR，adr，XR	subtract arithmetic
3	0				AND	GR，adr，XR	and
3	1				OR	GR，adr，XR	or
3	2				EOR	GR，adr，XR	exclusive or

4 0 4 1				CPA　GR, adr, XR CPL　GR, adr, XR	compare arithmetic compare logical
5 0 5 1 5 2 5 3				SLA　GR, adr, XR SRA　GR, adr, XR SLL　GR, adr, XR SRL　GR, adr, XR	shift left arithmetic shift right arithmetic shift left logical shift right logical
6 0 6 1 6 2 6 3 6 4	0 0 0 0 0			JPZ　adr, XR JMI　adr, XR JNZ　adr, XR JZE　adr, XR JMP　adr, XR	jump on plus or zero jump on minus jump on non zero jump on zero unconditional jump
7 0 7 1	0	 0	 0000	PUSH adr, XR POP　GR	push effective address pop up
8 0 8 1	0 0	 0	 0000	CALL adr, XR RET	call subroutine return from subroutine
9 ≀ F				未使用	

8．4　命令の種類と機能

本機はCASLで定義された23の命令を持っています。以下に各命令の機能を説明します。なお、説明には次の表記法を使います。

①GR　　　：GRの値を番号とする汎用レジスタを示します。（ただし、0≦GR≦4の範囲）

②XR　　　：XRの値を番号とする指標レジスタを示します。（ただし、1≦XR≦4の範囲）指標レジスタとしてGR1～GR4が使用できます。

③SP　　　：スタックポインタを示します。（スタックポインタはGR4を使用）

④adr　　　：ラベル名に対応する番地、または10進の定数（ただし、−32768≦adr≦65535）を示します。adrはアドレスとして0～65535の値を持ちますが、32768～65535の値を負の10進定数で記述することもできます。（例：65535番地は−1と記述できます。）

⑤有効アドレス：adrとXRの内容とのアドレス加算値、またはその値が示す番地です。

⑥（X）　　：X番地の内容、またはXが汎用レジスタ（GR）を示す場合は、そのGRの内容を表します。

⑦［ ］　　：［ ］に囲まれた部分は省略可能です。XRを省略した場合は、指標レジスタによる修飾は行われません。

【各命令と機能】

以下の説明においてAとはA番地に格納されている内容を示します。GR0～GR4においてもそれぞれのレジスタに格納されている内容（値）を示します。

(1)LD　　GR, adr [, XR]

機能：有効アドレスの内容をGRにロードします。

(2)ST　　GR, adr [, XR]

機能：有効アドレスにGRの内容を格納します。

(3)LEA　　GR, adr [, XR]

機能：有効アドレスそのものをGRにロードします。GRの値によりFRが設定されます。

〈例〉LEA　GR1,100　　　定数100をGR1に格納します。
　　　LEA　GR1,10,GR1　　GR1に10をたすことになります。
　　　LEA　GR1,0,GR2　　GR2の内容をGR1に移動させます。

(4)ADD　　GR, adr [, XR]

機能：有効アドレスの内容をGRに加算します。GRの値によりFRが設定されます。

(5)SUB　　GR, adr [, XR]

機能：有効アドレスの内容をGRから減算します。GRの値によりFRが設定されます。

〈例〉右図の内容のときは次のようになります。ただし、A番地には $(1010)_{16}$ が格納されているものとします。したがって、A+GR2は $(1015)_{16}$ になります。

〔実行前の状態〕

LD　GR1, A, GR2　$(1015)_{16}$ 番地の30をGR1にロード　　GR1 [15]
ST　GR1, A, GR2　GR1の15を $(1015)_{16}$ 番地に格納　　GR2 [5]
LEA　GR1, A, GR2　$(1015)_{16}$ をGR1に格納　　[20] $(1010)_{16}$ 番地
ADD　GR1, A, GR2　$(1015)_{16}$ 番地の30をGR1に加算
SUB　GR1, A, GR2　$(1015)_{16}$ 番地の30をGR1から減算　　[30] $(1015)_{16}$ 番地

(6)AND　　GR, adr [, XR]

(7)OR　　GR, adr [, XR]

(8)EOR　　GR, adr [, XR]

機能：1語16ビットのビットごとの論理積（AND）、論理和（OR）、排他的論理和（EOR）をGRの内容と有効アドレスの内容とで行い、その結果をGRに設定します。GRの値によりFRが設定されます。

〈例〉AND　GR1, MASK　MASK番地の内容を7にしておくと、GR1の下位3ビットが取り出せます。
　　　EOR　GR1, MASK　MASK番地の内容をFFFF（−1）にしておくと、GR1の全ビットが反転します。

(9)CPA　　GR, adr [, XR]

(10)CPL　　GR, adr [, XR]

機能：GRの内容と有効アドレスの内容との算術比較（CPA）または論理比較（CPL）を行います。比較結果によりFRに次の値を設定します。

比較結果	FRの設定ビット値
（GR）＞（有効アドレス）	00 (0)　正
（GR）＝（有効アドレス）	01 (1)　ゼロ
（GR）＜（有効アドレス）	10 (2)　負

（ ）はGRまたは有効アドレスの内容を示します。

算術比較：符号付の数として比較
論理比較：16ビットの整数として比較

〈例〉ＧＲ１の内容が$(-256)_{10}$でＡ番地の内容が$(256)_{10}$のときは

ＣＰＡ　ＧＲ１，Ａ　でＦＲは$(10)_2$になります。

ＣＰＬ　ＧＲ１，Ａ　でＦＲは$(00)_2$になります。

16ビット表記では$(-256)_{10}$は補数表示され$(FF00)_{16}$になり、$(256)_{10}$は$(0100)_{16}$になり、$(FF00)_{16}>(0100)_{16}$です。

⑾ＪＰＺ　adr［，ＸＲ］
⑿ＪＭＩ　adr［，ＸＲ］
⒀ＪＮＺ　adr［，ＸＲ］
⒁ＪＺＥ　adr［，ＸＲ］

機能：ＦＲの値により有効アドレスに分岐(ジャンプ)します。分岐しないときは次の命令に移ります。

命令	分岐するときのＦＲの値	意　　味
ＪＰＺ	00 または 01	正または０のとき分岐
ＪＭＩ	10	負のとき分岐
ＪＮＺ	00 または 10	正または負のとき分岐
ＪＺＥ	01	０のとき分岐

⒂ＪＭＰ　adr［，ＸＲ］

機能：ＦＲの値に関係なく無条件に有効アドレスに分岐(ジャンプ)します。

メモ　ＦＲレジスタについて

分岐命令はＦＲの値で分岐先を決めますが、ＦＲの値を決定する命令は次の命令です。
ＬＥＡ、ＡＤＤ、ＳＵＢ、ＡＮＤ、ＯＲ、ＥＯＲ、ＣＰＡ、ＣＰＬ、ＳＬＡ、ＳＲＡ、ＳＬＬ、ＳＲＬ

⒃ＳＬＡ　ＧＲ，adr［，ＸＲ］
⒄ＳＲＡ　ＧＲ，adr［，ＸＲ］

機能：符号ビット(最上位ビット)を除いたＧＲの内容を、有効アドレスで指定されたビット数だけ左(ＳＬＡ)または右(ＳＲＡ)へシフトします。シフトしてはみ出たビットは切り捨てられます。空いたビットには左シフトのときは０が入り、右シフトのときは符号ビットが入ります。シフトした後のＧＲの内容によりＦＲが設定されます。ＧＲが正のときはＦＲは$(00)_2$になります。

⒅ＳＬＬ　ＧＲ，adr［，ＸＲ］
⒆ＳＲＬ　ＧＲ，adr［，ＸＲ］

機能：符号ビットを含んだＧＲの内容を、有効アドレスで指定されたビット数だけ左(ＳＬＬ)または右(ＳＲＬ)へシフトします。シフトしてはみ出たビットは切り捨てられます。空いたビットには０が入ります。シフトした後のＧＲの内容によりＦＲが設定されます。ＧＲが正のときはＦＲは$(00)_2$になります。

〈例〉ＧＲ１＝1001 1100 0100 1011でＧＲ２＝３のとき

ＳＬＡ　ＧＲ１，３→３ビット左へシフトし　ＧＲ１＝1110 0010 0101 1000 になります。
（算術左シフト）

ＳＲＡ　ＧＲ１，３→３ビット右へシフトし　ＧＲ１＝1111 0011 1000 1001 になります。
（算術右シフト）

ＳＬＬ　ＧＲ１，１，ＧＲ２→４ビット左へシフトし　ＧＲ１＝1100 0100 1011 0000 になります。
（論理左シフト）

ＳＲＬ　ＧＲ１，１，ＧＲ２→４ビット右へシフトし　ＧＲ１＝0000 1001 1100 0100 になります。
（論理右シフト）

⒇ＰＵＳＨ　adr［，ＸＲ］

機能：ＳＰ(スタックポインタ)から１をアドレス減算した後、ＳＰで指定された番地に有効アドレスを格納します。

(21)ＰＯＰ　ＧＲ

機能：ＳＰで指定された番地の内容をＧＲに設定してから、ＳＰに１をアドレス加算します。

(22)ＣＡＬＬ　adr［，ＸＲ］

機能：有効アドレス(サブルーチン)に分岐する命令で、分岐する前にスタックに、戻り番地（次の命令の番地）を保持します。

(23)ＲＥＴ

機能：ＣＡＬＬに対する戻り命令で、スタックに保持されていた戻り番地により、メインプログラムに移ります。

8．5　アセンブラの文法

本機のアセンブラ言語はＣＡＳＬと同様に、疑似命令、マクロ命令、機械語命令を持っています。疑似命令、マクロ命令、機械語命令は次のように記述します。

ラベル欄	命令コード	オペランド欄	注釈欄
［ラベル］	ＳＴＡＲＴ	［実行開始番地］	［注釈］
空　白	ＥＮＤ	空　　白	［注釈］
［ラベル］	ＤＣ	定　　数	［注釈］
［ラベル］	ＤＳ	領域の語数	［注釈］
［ラベル］	ＩＮ	入力領域、入力文字長	［注釈］
［ラベル］	ＯＵＴ	出力領域、出力文字長	［注釈］
［ラベル］	ＥＸＩＴ	空　　白	［注釈］
［ラベル］	ＷＲＩＴＥ	空　　白	［注釈］
［ラベル］	機械語命令	命令の種類と機能　参照	［注釈］

表中の「空白」は記入してはならない箇所、［　］で囲まれた部分は省略可能を意味します。

8．6　疑似命令

疑似命令はアセンブラの制御、定数の定義、プログラムの連結のために必要なデータの生成などを行うもので、次の４種類があります。

⑴ＳＴＡＲＴ：実行開始番地の定義や、他のプログラムとの連結のための入り口名の定義をします。プログラムの最初に必要です。

⑵ＥＮＤ　：プログラムの終わりを定義します。プログラムの最後に必要です。

(3)DC　：define constant の略で、定数で指定したデータを格納します。
定数には次の4種があります。

①10進定数　　DC　n

nで指定した10進数値を1語の2進数データとして格納します。nは－32768～65535の範囲です。

②16進定数　　DC　#h

hは4桁の16進数（0～F）です。hで指定された16進数値を1語の2進数データとして格納します。#hは#0000～#FFFFの範囲です。

③文字定数　　DC '文字列'

文字列の左端から1文字ずつ連続する語の下位8ビット（第8～15ビット）に文字データを格納します。各語の第0～7ビットには0のビットが入ります。文字列には384ページのキャラクタ・コード表の内、コード32～38（&H20～&H26)、40～95（&H28～&H5F)、97～122（&H61～&H7A）および166～223（&HA6～&HDF）の文字が使えます。なお、文字列の長さは0であってはなりません。

④アドレス定数　　DC　ラベル名

ラベル名に対応するアドレス値を1語の2進数データとして格納します。

(4)DS　：define storage の略で、指定した語数の領域を確保します。
領域の語数は、10進定数（≧0）で指定します。0の場合、領域は確保されずラベル名だけが有効となります。

8．7　マクロ命令

(1) IN　入力領域，入力文字長：

キーボードから1レコードの文字データを入力領域のラベル名の番地に格納します。入力文字長のラベル名の番地には、入力した文字数が格納されます。

(2)OUT　出力領域，出力文字長：

出力領域に格納されている文字データを1レコードとして、表示またはプリンタに出力します。ただし、出力文字長のラベル名の番地で指定されている文字数分だけ出力します。

(3)EXIT　：プログラムの実行を終了します。

(4)WRITE：レジスタGR0～GR4およびPC、FRの内容を表示します。また、プリンタが印字できる状態になっているときは、GR0～GR3の内容を印字します。

〈例〉　　　　　　　　　　　　説　明

```
10      START
20      IN     A,C          キーボードで文字を入力
25      OUT    NL,N         入力表示と出力表示の区別のため、強制的に表示内容を変えている
30      OUT    A,B          キーボードで入力した文字の出力
40      EXIT
50A     DS     20           入力文字数を20に設定
60B     DC     2            出力文字数を2に設定
70C     DS     1            入力文字長を格納するアドレス
80N     DC     9           ┐
90NL    DC     'PPPPPPPPP  │ 強制表示内容の指定
  '                        ┘
100     END
```


［キー操作］

（アセンブル実行後）

G	↵	開始アドレスは#1000
N		
QWERTYUIOP	↵	PPPPPPPPP（強制表示）
	↵	QW
	↵	メニュー表示

①行番号30をOUT　A，Cに変えて実行してみてください。

②行番号25を削除して実行してみてください。

③行番号60の「2」を変えて実行してみてください。

いろいろ変えて実行することにより、その違いがわかります。

本機のマクロ命令では、次の命令群を生成します。

		＜オブジェクト＞		＜命令＞
IN	a，b	7000	aaaa	PUSH　a
		7000	bbbb	PUSH　b
		8000	0000	CALL　0
		1244	0002	LEA　4，2，4
OUT	a，b	7000	aaaa	PUSH　a
		7000	bbbb	PUSH　b
		8000	0002	CALL　2
		1244	0002	LEA　4，2，4
EXIT		6400	0004	JMP　4
WRITE		8000	0006	CALL　6

（注）aaaa、bbbbはラベルaまたはラベルbが定義されているアドレスを示します。

8．8　特殊命令

(1) [*]

機能：情報処理技術者試験の空欄穴埋め問題に対応するための、空欄入力命令です。

この命令をアセンブルすると#0000が2語生成されます。シミュレーションのときには命令語の第1語が#00XX（XXは何でもかまいません）の命令を実行後一時停止し、*STP*と表示します。このときオブジェクトの変更やレジスタ値の確認をすることができます。その後、プログラムを続行できます。

第８章
機械語モニタとアセンブラ機能

プログラム言語は用途により色々な種類があります。本機では、高級言語であるＢＡＳＩＣや機械語も使用できます。
ＢＡＳＩＣは英語に近い形で使いやすい言語ですが、処理スピードの点では機械語に劣ります。一方、機械語は人間にとって非常にわかりにくい言語ですが、いろいろなことができ、コンピュータの機能を最大限に活用できます。
本機はこの機械語を使用するために、機械語モニタとアセンブラ機能が内蔵されています。
機械語モニタ機能では、メモリ内の機械語の表示、メモリ内の書き換え、機械語の実行などが簡単な命令だけで行えます。
アセンブラ機能は、ニモニック（アセンブラ言語）で書かれたソースプログラムを機械語に変換する機能です。
このポケットコンピュータはＣＰＵ（シーピーユー：中央演算処理装置）に、優れた８ビットＣＰＵとして広く利用されているＺ80（ＣＭＯＳ　Ｚ80相当品）を使用しています。
Ｚ80は人気のあるＣＰＵであり、機械語やニモニックについての入門書等の関連書籍が市販されていますので、それらの書籍を参照してください。

この章では、まず機械語モニタ機能の各命令（コマンド）の働きについて説明し、その後ソースプログラムの作成、アセンブルのしかたなどを説明します。

機械語を使う場合のご注意

１．機械語を実行したときに、プログラムや操作に誤りがあると、次のようなことが起こります。

①プログラムを実行し続けて、すべてのキーが働かなくなる。
　この場合は、リセットスイッチを押します。
②でたらめな表示や動作を続ける。
　この場合は、BREAK を押します。それでも止まらないときは、リセットスイッチを押します。
③プログラムやデータの一部またはすべてが、壊れたり、消えたりする。
　これは機械語プログラムだけでなく、他のプログラム、ファイル、データに及ぶことがあります。

これらが起こったときは、計算機の内部の状態がどのようになっているかわかりませんので、リセットスイッチを押し、すべての内容を消去してください。

ご注意
機械語プログラムを扱うときは、上記のようなことが起こる可能性があります。消えては困るプログラムやデータは、紙に書き写しておいてください。プリンタをお持ちの場合は、印字しておいてください。

２．確保した機械語エリアだけを使用してください。

機械語モニタのＵＳＥＲコマンドで確保した機械語エリア以外を使用すると、ＢＡＳＩＣやＴＥＸＴのプログラムなど、ほかの記憶内容を壊したり、正しい動作をしないことがあります。

参　考

機械語関係にはむずかしい用語がありますので、主な用語をまとめておきます。

機械語　コンピュータが直接、解読・実行できる言語です。16進法で表されます（コンピュータ内では２進数で扱われます）。

ニモニック　機械語コードを人間が記憶しやすいように符号化したものです。
たとえば加算命令（addition）のコードをＡＤＤのように定めます。
ニモニックを言語的に体系付けたものをアセンブラ（アセンブリ）言語と呼びます。

ソースプログラム
ニモニック（アセンブラ言語）で書かれたプログラム。機械語プログラムに変換する元になるプログラムです。

オブジェクトプログラム
変換して得られたプログラムで、そのまま計算機で実行可能な形にされたものです。一般的には、ソースプログラムをアセンブルして得られた機械語プログラムのことです。単に**オブジェクト**と呼ぶこともあります。（オブジェクトは、機械語のコード１つ１つを指す場合と機械語プログラム全体を指す場合があります。）

アセンブル　ソースプログラムを機械語プログラムに変換（翻訳）することです。
人間が人手によりアセンブルを行うことを**ハンドアセンブル**と呼びます。

アセンブラ　アセンブルをコンピュータに行わせるための翻訳プログラムのことです。

疑似命令　オブジェクトを格納する場所（アドレス）を指定したり、データを生成させたりするアセンブラ制御用の命令です。それ自体は機械語に変換されません。

生成　作り出すこと。ニモニックからオブジェクトを作り出すことを言います。

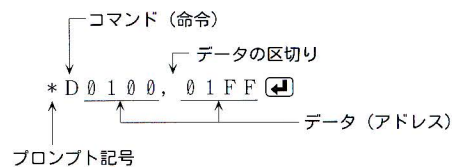
アドレス　メモリにつけられている番号をアドレス（番地）と呼びます。
機械語では、メモリを使用するときアドレスを指定して使用します。

機械語モニタ機能

１．機械語モニタを使ううえでのきまり

機械語モニタは機械語モニタモードで使います。ＢＡＳＩＣモード（ＲＵＮまたはＰＲＯモード）でＭＯＮ ↵ と押せば、機械語モニタモードに切り替わります。
右の画面が表示されます。
（パスワードが設定されているときは、機械語モニタモードにできません。）

```
MACHINE LANGUAGE MONITOR
*
```

＊記号は機械語モニタモードのプロンプト記号です。
コマンド（命令）は必ずこのプロンプト記号の後に書き、アドレス（番地）やデータの指定が必要な場合は、コマンドに続けて指定します。そして[↵]を押して、実行させます。
〈例〉

● アドレスおよびデータの指定は16進数で行います。
● アドレスおよびデータの区切りはコンマ（，）で指定します。
● アドレスおよびデータの指定に０～Ｆの16進数字とコンマ以外を入れるとエラー（ＳＹＮＴＡＸ　ＥＲＲＯＲ）になります。
● 機械語モニタモードは、[BASIC]、[TEXT]の操作または電源を切ることで解除されます。

２．機械語モニタの各命令の説明

＊ＵＳＥＲ……ユーザーエリア

機　能　機械語エリアの確保および確保したエリアの表示を行います。
書　式　(1)　ＵＳＥＲ 終了番地[↵]
(2)　ＵＳＥＲ[↵]
(3)　ＵＳＥＲ ００ＦＦ[↵]
説　明　● 書式(1)を実行するとメモリの０１００Ｈ番地（開始番地）から指定した終了番地までを機械語エリアとして確保します。開始番地は自動的に０１００Ｈ番地になります。
ＵＳＥＲ０１ＦＦ[↵]と実行した例

● 書式(2)を実行すると、現在確保されている機械語エリアの番地を表示します。

```
＊ＵＳＥＲ
ＦＲＥＥ：０１００－０１ＦＦ
＊
```

機械語エリアが確保されていないときは“ＦＲＥＥ：ＮＯＴ　ＲＥＳＥＲＶＥＤ”と表示されます。
● 書式(3)を実行すると、機械語エリアを消去します。そして、次の表示をします。
ＵＳＥＲ００ＦＦ[↵]
“ＦＲＥＥ：ＮＯＴ　ＲＥＳＥＲＶＥＤ”
● 確保できない範囲（機械語エリアとして使用できない範囲）を指定するとエラー（ＭＥＭＯＲＹ　ＥＲＲＯＲ）になります。

＊Ｓ……………セットメモリ

機　能　メモリの内容を書き換えます。
書　式　(1)　Ｓ 開始番地[↵]
(2)　Ｓ[↵]
説　明　● 書式(1)を実行すると、指定した開始番地の現在のメモリ内容が表示され、変更するデータの入力待ちになります。
Ｓ０１００[↵]と実行した例

● データを変更するときは、１バイトデータ（２桁の16進の数字）を入力し、[↵]を押します。データが変更され、次のアドレス（番地）のデータ入力待ちになります。
データを変更しない場合は、データを入力せずに[↵]のみを押します。そのまま次のアドレスのデータ入力待ちになります。
● データは２桁以上入力できません。入力途中の値を取り消すときは[◀]または[CLS]を押します。
● [▼]で、次のアドレスを呼び出すことができます。また[▲]で前のアドレスを呼び出すことができます。
● 書式(2)を実行すると、以前のＳコマンドで表示していた次のアドレスのメモリ内容が呼び出されます。電源を入れた後（初期値）では０１００の指定になります。
● [BREAK]を押せばコマンド（命令）待ちの状態に戻ります。
〈例〉次のオブジェクトプログラムを０１００Ｈ番地から書き込みます。

```
〈オブジェクトプログラム〉
３Ｅ　０１
１８　０４
３Ａ　０Ｆ　０１
３Ｃ
３２　０Ｆ　０１
Ｃ９
```

＝参考＝　上記オブジェクトプログラムのソースプログラムを示します。

```
〈ソースプログラム〉
       ＬＤ    Ａ，０１Ｈ         ←Ａレジスタに０１Ｈを入れる
       ＪＲ    ＳＴ               ←ラベルＳＴの番地へジャンプ
       ＬＤ    Ａ，（０１０ＦＨ）  ←Ａレジスタに０１０ＦＨ番地の内容を入れる
       ＩＮＣ  Ａ                 ←Ａレジスタの内容に１を加算する
ＳＴ： ＬＤ    （０１０ＦＨ），Ａ  ←Ａレジスタの内容を０１０ＦＨ番地へ入れる
       ＲＥＴ                     ←リターン
```

［キー操作］	［表　示］	
BREAK		
ＵＳＥＲ０１ＦＦ↵	＊ＵＳＥＲ０１ＦＦ	０１００Ｈ番地から０１ＦＦＨ番地までを機械語エリアとして確保（余裕のある大きさです）
	ＦＲＥＥ：０１００－０１ＦＦ	
Ｓ０１００↵	＊Ｓ０１００	
３Ｅ↵	０１００：１０－３Ｅ	
０１↵	０１０１：１２－０１	
１８↵	０１０２：００－１８	
０４↵	０１０３：００－０４	
３Ａ↵	０１０４：０１－３Ａ	
０Ｆ↵	０１０５：１０－０Ｆ	
０１↵	０１０６：５０－０１	
３Ｃ↵	０１０７：４３－３Ｃ	
３２↵	０１０８：２Ｄ－３２	
０Ｆ↵	０１０９：４７－０Ｆ	
０１↵	０１０Ａ：３８－０１	
Ｃ９↵	０１０Ｂ：３０－Ｃ９	
BREAK	０１０Ｃ：３１－	
	＊	

ここには以前の内容が表示されるため、このとおりの数値になるとは限りません。

＊Ｄ……………ダンプメモリ

機　能　メモリの内容を表示させます。

書　式　(1)　Ｄ 開始番地↵

(2)　Ｄ↵

(3)　Ｄ 開始番地，終了番地↵

説　明　● 書式(1)を実行すると、開始番地から24バイト分（６行分）のメモリの内容が表示されます。（プリントモードでは、開始番地から16バイト分（４行分）を印字します。）

Ｄ０１００↵と実行した例　　**〈表示例〉**

```
16バイト単位の先頭アドレス→  0100  :  3E  01  18  04  >...
            チェックサム→  (1D)      3A  0F  01  3C  :..<
                                    32  0F  01  C9  2../
                                    31  00  00  00  1...
```

０１０Ｃ番地以降は前に記憶していた内容が残っています。そのためチェックサムもこのようになるとは限りません。

各データをアスキーコードとした場合の該当文字を表示します。ただし、００Ｈ～１ＦＨ、Ｆ９Ｈ～ＦＦＨのデータはピリオド（.）で表示されます。

● 開始番地、終了番地のアドレス（番地）の区分は規定（ＸＸＸ０Ｈ～ＸＸＸＦＨ番地）されていて、その区分内であればどのアドレスを指定しても同じになります。たとえば０１

０４Ｈ番地を指定した場合は、０１００Ｈ～０１１７Ｈ番地の内容を表示します。（０１００Ｈ～０１０ＦＨの16バイト分を単位として、この16バイト分を含む24バイト分の内容を表示します。）

● ▼を押せば、次の単位の16バイト分からのメモリ内容が表示されます。▲を押せば、前の単位の16バイト分からのメモリ内容が表示されます。

● 書式(2)を実行すると、以前にＤコマンドで表示していた次の単位の先頭番地から表示します。初期値は０１００Ｈになります。プリントモード時は印字します。

● 書式(3)を実行すると、プリントモード時は開始番地～終了番地のブロックのメモリ内容を印字します。印字が終了すれば、コマンド待ちの状態に戻ります。

プリントモードでないときは、開始番地から24バイト分のメモリの内容を表示します。（この場合、終了番地の指定は意味がなくなります。）

● プリントモードの指定／解除は、Ｐコマンドまたは SHIFT ＋ P⇔NP で行います。

● BREAK を押せばコマンド（命令）待ちの状態に戻ります。

チェックサム…**検査合計**　データ項目の集まりの合計であって、そのデータが記録されるとき計算され、検査の目的でその集まりにつけられるもの。本機では、Ｄコマンドの実行によって表示された16バイト分のメモリの内容を合計し、その合計値の下位１バイトをチェックサムとして表示します。

プログラムをマニュアル操作で入力したときなど、このチェックサムが元のプログラムと一致することを確認することにより、表示された16バイトの中に入力誤りがないかどうかを確認できます。ただし、入力誤りが２カ所以上あった場合は、偶然チェックサムが一致してしまうことがあります。

＊Ｅ……………イクスチェンジメモリ

機　能　メモリの内容を書き換えます。

書　式　(1)　Ｅ 開始番地↵

(2)　Ｅ↵

説　明　● メモリ内容の書き換えは、Ｓコマンドでも行えますが、Ｅコマンドを使用すると便利に書き換えができます。

● 書式(1)を実行すると、指定した開始番地を含む24バイト分を表示し、指定した番地でカーソルが点滅します。指定できる範囲は００００Ｈ～７ＦＦＦＨまでです。

● カーソル移動キー（◀▶▲▼）でカーソルを移動させ、16進数で入力を行います。

データを入力すると、自動的に次の番地にカーソルが移動しますので、続けて入力できます。

● 16進数のＡ～Ｆはアルファベットキーで入力しますが、右のように四則演算キーでも入力できます。

7	8	9	／ (F)
4	5	6	＊ (E)
1	2	3	－ (D)
0	. (A)	＝ (B)	＋ (C)

● TAB を押すとカーソルは右側の文字表示に移動し、直接データをアスキー文字(数字および英文字など)で入力することができます。もう一度 TAB を押すと、左側の16進数でのコード入力に移ります。

● 書式(2)を実行すると、以前にＥコマンドで表示していた次の単位の先頭番地から表示します。初期値は０１００Ｈになります。

● Ｅコマンドを実行すると“カナ”モードは解除されます。

＊Ｐ……………プリントスイッチ

機　能　プリントモードの設定／解除を行います。

書　式　Ｐ ↵

説　明　● Ｐ ↵ と押すたびに、プリントモードの設定と解除が交互に行われます。(プリントモードでは画面右下に“ＰＲＩＮＴ”が点灯します。)

SHIFT ＋ P↔NP の操作と同じです。

● プリンタが接続されていないときや、プリンタの電源が入っていないときは、Ｐコマンドは働きません。

＊Ｇ……………ゴーサブ

機　能　指定した開始番地から機械語プログラムを実行します。

書　式　Ｇ 開始番地 ↵

説　明　● ＢＡＳＩＣのＧＯＳＵＢ命令と同様で、指定した開始番地から機械語プログラムを実行し、機械語のリターン命令があればコマンド待ちの状態に戻ります。

● 必ずプログラムの最後にＲＥＴ（リターン）命令を入れてください。

ＲＥＴ命令がないと暴走します。

暴走…でたらめな実行、または一定の実行を続けて止まらなくなること。内部の状態によって、どのような動きをするか分からず、多くの場合、機械語プログラムやＢＡＳＩＣプログラム、データなど記憶内容を破壊します。

● プログラムの実行を中止する場合は、リセットスイッチを押します。

（注）機械語プログラムは１カ所でもまちがいがあると暴走することがあります。このため、ＢＡＳＩＣなど、他のプログラムが計算機に入っているときは、あらかじめ紙に書き写したり、印字したりしておいてください。

〈例〉Ｓコマンドの例で入力したプログラムを実行する場合

Ｇ０１００ ↵

```
*G0100
*
```

すぐにコマンド待ちの状態に戻ります。

０１０ＦＨ番地に０１Ｈが入っていますので、Ｄコマンドで確認します。

Ｄ０１００ ↵

```
0100 :  3E  01  18  04  >...
(1E)    3A  0F  01  3C  :..<
        32  0F  01  C9  2../
        31  00  00  01  1...
```

↑
０１Ｈが入っています

次に０１０４Ｈから実行します。

BREAK Ｇ０１０４ ↵

０１０ＦＨ番地に１が加えられています。Ｄコマンドで確認します。

Ｄ０１００ ↵

↑
１が加えられています

繰り返し０１０４Ｈ番地から実行すれば、その都度０１０ＦＨ番地に１が加えられます。

＊Ｒ……………リードＳＩＯ

機　能　ＳＩＯ（シリアル入出力装置）に送られてくるデータを読み込みます。パソコン等との機械語の通信等に使用します。

書　式　(1)　Ｒ ↵

(2)　Ｒ 開始番地 ↵

説　明　ＳＩＯからインテル・ヘキサ形式で送られてくるデータを読み込みます。

● 書式(1)では、Ｗ命令実行時の番地に読み込みます。

● 書式(2)は読み込んだデータを指定した開始番地から順番に入れていきます。

● 読み込みが終了すると、データが入ったエリア（領域）を表示します。

● 読み込みを中止するときは、コマンド待ちの状態になるまで、BREAK を押し続けてください。

● 通信条件の設定は、ＴＥＸＴモードのＳＩＯで行ってください。

＊Ｗ……………ライトＳＩＯ

機　能　ＳＩＯ（シリアル入出力装置）にデータを出力します。パソコン等との機械語の通信等に使用します。

書　式　Ｗ 開始番地，終了番地 ↵

説　明　● 開始番地から終了番地までのメモリ内容をＳＩＯからインテル・ヘキサ形式で出力します。

● 出力を中止するときは、コマンド待ちの状態になるまで、BREAK を押し続けてください。

● 通信条件の設定は、ＴＥＸＴモードのＳＩＯで行ってください。

（注）周辺機器接続端子（11ピン）にプリンタを動作状態で接続し、このＷコマンドを実行すると本機およびプリンタが誤動作することがあります。この場合は、プリンタの電源を切り、本機の BREAK を押し続けてください。

＊ＢＰ………ブレークポイント

機　能　指定したアドレス（番地）にブレークポイントを設定します。

書　式　(1)　ＢＰ 番地［－カウント数］↵

(2)　ＢＰ↵

(3)　ＢＰ　0↵

説　明

● 書式(1)を実行すると、指定した番地にブレークポイントが設定されます。
ブレークポイントは４カ所まで設定できます。４カ所設定するときは、書式(1)と同様の方法で、１カ所ずつアドレスを設定します。指定できる番地の範囲は０００００Ｈ～７ＦＦＦＨです。

● カウント数は０～２５５の範囲で指定できますが、０を指定すると、ブレークポイントの解除になります。カウント数を省略した場合は１の指定とみなします。

● 同じ番地でもカウント数が異なる場合は、別々のブレークポイントとみなします。
カウント数を０として解除した場合は、同じ番地のブレークポイントすべてを解除します。

● ブレークポイントを同時に設定できるのは４ヵ所までです。５カ所以上設定しようとすると、先に設定したブレークポイントから解除されます。

● ブレークポイントは命令（オペコード）の番地に設定してください。オペランドの番地に設定すると、実行が停止しないだけでなく、プログラムが正しく実行されません。

● 書式(2)では、設定されているブレークポイントのアドレスとカウント数を表示します。設定されていない場合は、次の行にプロンプト（＊）だけを表示します。

● 書式(3)では、設定されているブレークポイントすべてを解除します。特定の番地のみ解除するときは、書式(1)でカウント数を０にします。

● １つのブレークポイントは、１回実行すると無効になります。したがって、繰り返しループの中では、指定したカウント数のときにブレークポイントとして１回だけ有効になります。
ただし、Ｇコマンドを実行すれば、再び指定したカウント数のときに有効になります。

● 機械語モニタ以外のモードから、機械語モニタモードにしたときでも、以前に設定したブレークポイントは記憶されており、Ｇコマンドを実行すると有効になります。

〈例〉次の機械語プログラムが０１００Ｈ～０１０ＤＨ番地に格納されているものとして、ブレークポイントを０１０５Ｈ番地と０１０６Ｈ番地に設定して実行してみましょう。（ブレークポイントは、命令がある番地に設定します。）

〈アドレス〉	〈機械語コード〉	〈ニモニック〉		〈プログラムの意味〉
０１００	３Ｅ		LD　A, 20H	Ａレジスタに２０Ｈ（16進数）を入れる
０１０１	２０			
０１０２	２１		LD　HL, 0400H	ＨＬレジスタに０４００Ｈを入れる
０１０３	００			
０１０４	０４			
０１０５	７７	LBL：	LD　(HL), A	ＨＬの内容で示す番地のメモリにＡの内容を入れる
０１０６	３Ｃ		INC　A	Ａの内容に１を加えてＡに入れる
０１０７	２３		INC　HL	ＨＬの内容に１を加えてＨＬに入れる
０１０８	ＦＥ		CP　0A0H	Ａの内容とＡ０Ｈを比較（Ａの内容からＡ０Ｈを減算）
０１０９	Ａ０			
０１０Ａ	Ｃ２		JP　NZ, LBL	前の計算で結果が０でない（Ａの内容がＡ０Ｈでない）とき、ラベルＬＢＬのある番地へジャンプ
０１０Ｂ	０５			
０１０Ｃ	０１			
０１０Ｄ	Ｃ９		RET	リターン

機械語モニタモードにして、０１０５Ｈ番地にブレークポイントを設定します。

BASIC ＭＯＮ↵

ＢＰ０１０５↵

```
MACHINE LANGUAGE MONITOR
*BP0105
BP=0105(   1)
*
```

０１０６Ｈ番地にブレークポイントを設定します。

ＢＰ０１０６－３↵

```
MACHINE LANGUAGE MONITOR
*BP0105
BP=0105(   1)
*BP0106-3
BP=0105(   1), 0106(   3)
*
```

ブレークポイントが２カ所に設定されました。

次にＧコマンドで０１００Ｈ番地から実行します。

Ｇ０１００↵

```
PC=0105   AF=20 44
SP=7FF6   BC=00 00
IX=7C06   DE=00 00
IY=7C03   HL=04 00
```

ＰＣ（プログラムカウンタ）の値が、停止したアドレスを示します。
Ａレジスタに２０Ｈが入っています。

ブレークポイントを設定したアドレスで実行を停止し、レジスタの内容を表示します。

実行を再開させます。

↵

```
PC=0106   AF=22 87
SP=7FF6   BC=00 00
IX=7C06   DE=00 00
IY=7C03   HL=04 02
```

０１０６Ｈ番地で停止。
Ａレジスタには２回カウントアップした結果２２Ｈが入っています。

２カ所目のブレークポイント設定アドレスで停止します。

このように、機械語プログラムを変更することなく、ブレークポイントの設定ができます。

（注）・ブレークポイントを設定すると、設定アドレスの内容を一時Ｆ７Ｈに置き換えてプログラムを実行します。
ブレークポイントを設定した状態で、プログラム実行中にリセットスイッチ（ＲＥＳＥＴ）を押すと、そのとき実行していないブレークポイント設定アドレスのメモリ内容が、Ｆ７Ｈに置き換えられたままになります。このような場合は、ブレークポイントを設定したアドレスの内容を確認して、Ｆ７Ｈになっているときは元の値に書き直す必要があります。

・ブレークポイントが不要になったときは、書式(3)（ＢＰ０↵）を実行して解除してください。
設定されたブレークポイントは、モードを切り替えたり、機械語プログラムを書き換えても保

持されています。そして、Ｇコマンドで機械語プログラムを実行するたびに、設定アドレスで停止します。

・メモリ内容がＦ７Ｈのアドレスには、ブレークポイントを設定できません。

３．機械語モニタモードでのエラー表示

機械語モニタモードでは以下のエラー表示があらわれます。

エラーメッセージ	内　　容
ＳＹＮＴＡＸ　ＥＲＲＯＲ	文法的に実行できない。
ＭＥＭＯＲＹ　ＥＲＲＯＲ	機械語エリアとして使用できる範囲を外れて、機械語エリアを確保しようとした。
Ｉ／Ｏ　ＤＥＶＩＣＥ　ＥＲＲＯＲ	ＳＩＯに対するリードインエラー、チェックサムエラーなど。
ＯＴＨＥＲ　ＥＲＲＯＲ	その他のエラー。

アセンブラ機能

１．ソースプログラムと機械語プログラム

機械語は、コンピュータが直接、解読・実行できる唯一の言語です。
機械語でプログラムを作れば、ＢＡＳＩＣなどに比べて処理スピードが速くなり、コンピュータの機能を最大限に活用できます。
しかし、この機械語はすべて数字で表され、人間にとっては大変覚えにくく、扱いにくい言語です。このため、機械語のプログラムを作成するときは、先に各命令を覚えやすい符号に置き換えて（符号化して）プログラムを作成した後、符号を機械語コードに変換して機械語プログラムを完成させる方法が用いられます。

符号化した命令（ニモニック）で書かれたプログラムをソースプログラムと呼びます。
これに対して、変換して得られた機械語プログラムをオブジェクトプログラムと呼びます。（274ページの参考を参照）

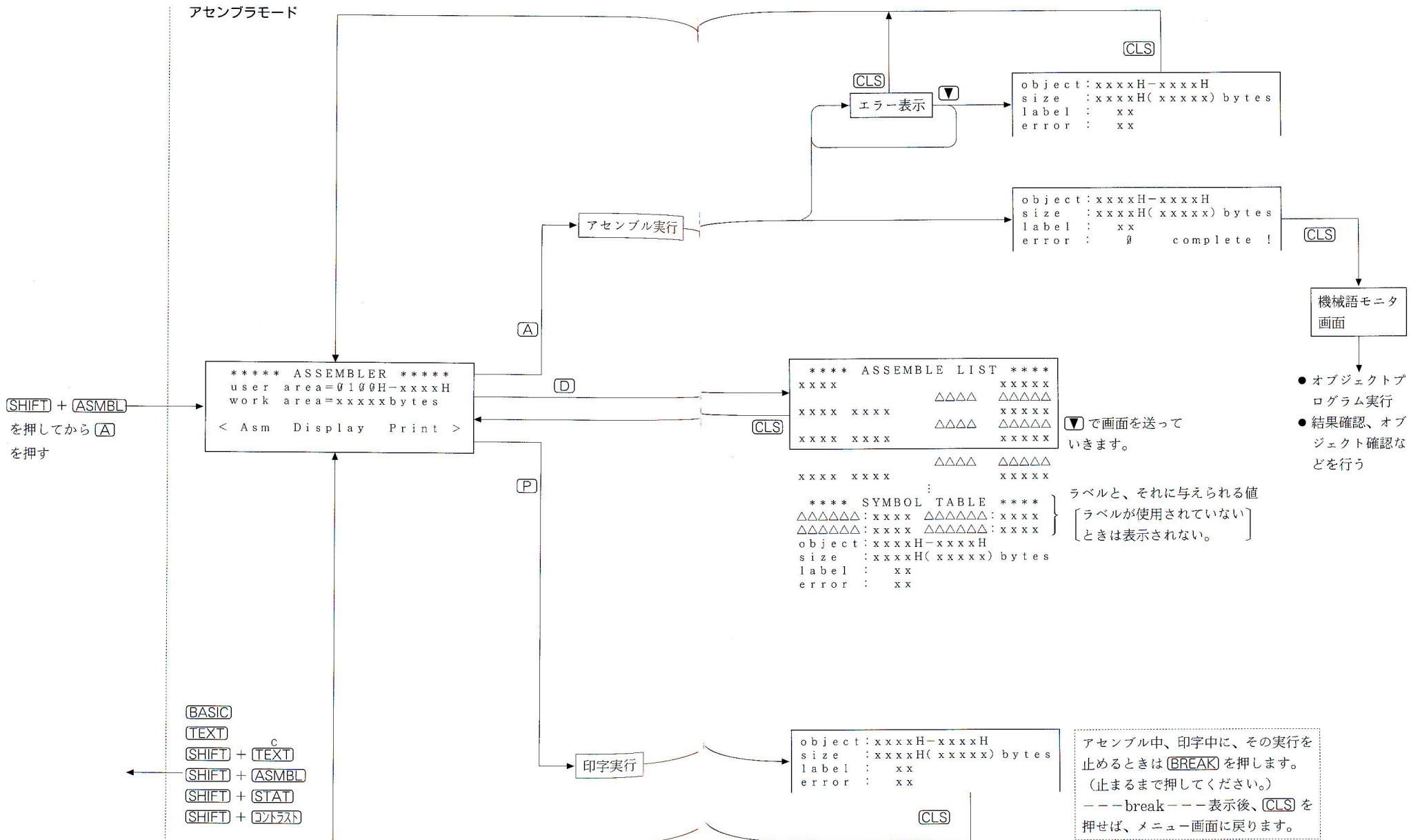
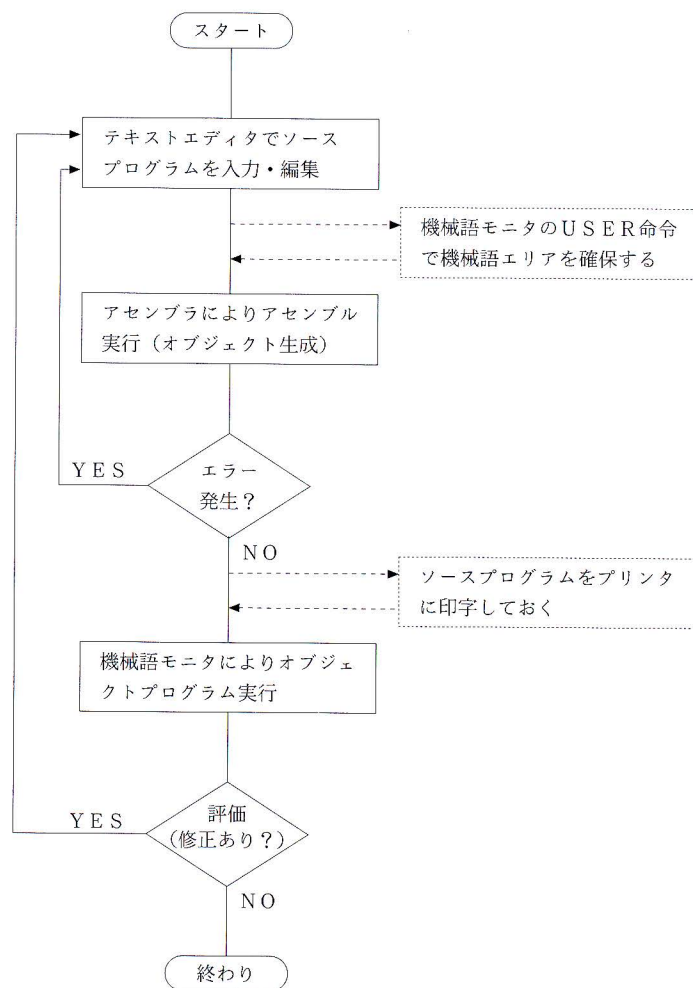
〈例〉

〈ニモニックで書かれたプログラム〉（ソースプログラム）	〈機械語プログラム〉（オブジェクトプログラム）	〈プログラムの意味〉
ＬＤ　Ａ，２	３Ｅ　０２	Ａレジスタに２を入れなさい
ＬＤ　Ｂ，３	０６　０３	Ｂレジスタに３を入れなさい
ＡＤＤ　Ａ，Ｂ	８０	ＡとＢの内容を加えて、Ａレジスタに入れなさい
ＬＤ　（０１０ＦＨ），Ａ	３２　０Ｆ　０１	０１０ＦＨ番地のメモリにＡレジスタの内容を入れなさい
ＲＥＴ	Ｃ９	戻りなさい（サブルーチン・リターン）

この例のように、ニモニックで書かれたプログラムは、機械語プログラムに比べるとわかりやすいので、プログラムが作りやすくなります。

ニモニックで書かれたプログラムをコンピュータで実行させるためには、機械語プログラムに変換しなければなりません。この変換作業をアセンブルと呼びます。
人間が手操作でアセンブルすることをハンドアセンブルと呼びます。上例のような短いプログラムなら、ハンドアセンブルしてもすぐに終わりますが、長いプログラムをハンドアセンブルするのは大変な作業で、まちがえることもあります。そこで登場するのが、アセンブラです。
アセンブラは、ニモニックで書かれたプログラムを機械語プログラムに翻訳（変換）するプログラムです。
これを使えば大変速く、アセンブルするときのまちがいがなくなります。
本機は、Ｚ80機械語用のアセンブラを持っています。以降に、このアセンブラの使いかたを説明します。

2．アセンブラの機能

アセンブラ機能一覧

アセンブラの手順

３．アセンブルしてみましょう

アセンブラやソースプログラムを説明する前に、とりあえずプログラム例を入れて、アセンブルしてみましょう。そして、できたオブジェクトプログラム（機械語プログラム）を実行してみましょう。

機械語プログラムを実行する前に知っておいていただきたいことがあります。
機械語プログラムを実行したとき、プログラムや操作に誤りがあると、すべてのプログラムやデータが消えることがあります（273ページ参照）。したがって、**消えては困るプログラムやデータは、紙に書き写したり、印字しておいてください。**

〈プログラム例〉
０４００Ｈ～０４７ＦＨ番地のメモリに２０Ｈから９ＦＨまでの数値を入れるプログラム（Ｈは16進数を示す記号（文字）です)。

〈ソースプログラム〉			〈プログラムの意味〉
１０	ＯＲＧ	０１００Ｈ	（オブジェクトを０１００Ｈ番地以降に入れなさい）
２０ＳＴＡＲＴ：	ＬＤ	Ａ，２０Ｈ	Ａレジスタに２０Ｈを入れる
３０	ＬＤ	ＨＬ，０４００Ｈ	ＨＬレジスタに０４００Ｈを入れる
４０ＬＢＬ：	ＬＤ	（ＨＬ），Ａ	ＨＬの内容で示す番地のメモリにＡの内容を入れる
５０	ＩＮＣ	Ａ	Ａの内容に１を加えてＡに入れる
６０	ＩＮＣ	ＨＬ	ＨＬの内容に１を加えてＨＬに入れる
７０	ＣＰ	０Ａ０Ｈ	Ａの内容とＡ０Ｈを比較（Ａの内容からＡ０Ｈを減算）
８０	ＪＰ	ＮＺ，ＬＢＬ	前の計算で結果が０でない（Ａの内容がＡ０Ｈでない）とき、ラベルＬＢＬのある番地へジャンプ
９０	ＲＥＴ		リターン（サブルーチンリターン）
１００	ＥＮＤ		（ソースプログラム終わり）

〈オブジェクト〉
```
3E  20
21  00  04
77
3C
23
FE  A0
C2  05  01   ←ラベルは番地に変換されます。（０１０５Ｈ番地）
C9
```

ソースプログラムの10行と100行の命令は、疑似命令と呼ばれ、アセンブラを制御する命令です。それ自身は機械語（オブジェクト）に変換されません。(300ページ参照)

(1)ソースプログラムの入力

それでは、ソースプログラムを入力してみましょう。ソースプログラムはＴＥＸＴプログラムとして入力します。（169ページ参照）

[TEXT] を押してテキストモードの機能選択画面（メインメニュー画面）にします。

```
 *** TEXT EDITOR ***

 Edit Del  Print
 Sio  File Basic Rfile
```

[E] を押して、エディット機能を選びます。

```
TEXT EDITOR
<
```

すでにＴＥＸＴプログラムが入っているときは、機能選択画面で [D] を押した後、[Y] を押して消去してください。

ソースプログラムを入力します。

［キー操作］	［表　示］		
１０[TAB]ＯＲＧ[TAB]０１００Ｈ[↵]	１０	ＯＲＧ	０１００Ｈ
２０ＳＴＡＲＴ：ＬＤ[TAB]Ａ，２０Ｈ[↵]	２０ＳＴＡＲＴ：	ＬＤ	Ａ，２０Ｈ
３０[TAB]ＬＤ[TAB]ＨＬ，０４００Ｈ[↵]	３０	ＬＤ	ＨＬ，０４００Ｈ
４０ＬＢＬ：[TAB]ＬＤ[TAB]（ＨＬ），Ａ[↵]	４０ＬＢＬ：	ＬＤ	（ＨＬ），Ａ
５０[TAB]ＩＮＣ[TAB]Ａ[↵]	５０	ＩＮＣ	Ａ
６０[TAB]ＩＮＣ[TAB]ＨＬ[↵]	６０	ＩＮＣ	ＨＬ
７０[TAB]ＣＰ[TAB]０Ａ０Ｈ[↵]	７０	ＣＰ	０Ａ０Ｈ
８０[TAB]ＪＰ[TAB]ＮＺ，ＬＢＬ[↵]	８０	ＪＰ	ＮＺ，ＬＢＬ
９０[TAB]ＲＥＴ[↵]	９０	ＲＥＴ	
１００[TAB]ＥＮＤ[↵]	１００	ＥＮＤ	

入力が終わったら、まちがいがないことを確認してください。

アセンブルの前にオブジェクトを格納するための機械語エリアを確保しておきます。確保しておかないとアセンブルできません。

(2)機械語エリアの確保

機械語エリアは機械語モニタのＵＳＥＲ命令で確保します。（275ページを参照）

機械語モニタモードにします。

[BASIC] ＭＯＮ [↵]

```
MACHINE LANGUAGE MONITOR
*
```

ＵＳＥＲ命令で機械語エリアを確保します。

ここでは、０４ＦＦＨまで（０１００Ｈ～０４ＦＦＨ）を確保しておきます。

ＵＳＥＲ０４ＦＦ [↵]

```
MACHINE LANGUAGE MONITOR
*USER04FF
FREE:0100-04FF
*
```


確保した機械語エリア（ユーザーエリア）が表示されます。

(3)アセンブルの実行

ソースプログラムを機械語プログラムに変換します。

[SHIFT] ＋ [ASMBL] と押してから [A]

```
 ***** ASSEMBLER *****
 user area=0100H-04FFH
 work area= 4645bytes

< Asm   Display   Print >
```

アセンブラモードになります。

（画面のワークエリアの値は変わることがあります。295ページを参照してください。）

[A]

アセンブルが実行されます。

```
 ***** ASSEMBLER *****


   --- assembling ---
```

アセンブルが正常に終われば右のように表示されます。（画面の内容については296ページ参照）

```
object:0100H-010DH
size  :000EH(    14) bytes
label :    2
error :    0    complete !
```

アセンブル中にエラーが発生したときは、右のようにエラーの内容と、エラーが発生した行が表示されます。（296ページ参照）

このときはテキストエディタへ戻ってソースプログラムを修正します。

```
 ***** ASSEMBLER *****

*FORMAT ERROR         (1)
0105 ****              40
       LBL:  LD     HL),A
```

(4)生成したオブジェクトの確認

生成したオブジェクトを機械語モニタで確認してみましょう。オブジェクトは０１００Ｈ～０１０ＤＨ番地に入っています。

[CLS] を押します。

（または [BASIC] ＭＯＮ [↵]）

機械語モニタモードになります。

Ｄ命令（ダンプメモリ）で呼び出します。

Ｄ０１００ [↵]

オブジェクトが表示されます。

```
MACHINE LANGUAGE MONITOR
*
```

```
0100 :  3E 20 21 00   >  !.
(88)    04 77 3C 23   . w<#
        FE A0 C2 05   .  ツ.
        01 C9 00 00   . ノ..
0110 :  20 00 19 00     ...
        00 00 00 00   ....
```

（０１０ＥＨ番地以降は前に記憶していた内容が残っています。）

(5)オブジェクト（機械語）プログラムの実行

生成したオブジェクトプログラムを実行してみましょう。機械語モニタのＧ命令（ゴーサブ）で実行します。

機械語モニタの命令待ち状態に戻します。

BREAK

```
*
```

実行します。

Ｇ０１００ ↵

実行が終わると命令待ち状態に戻ります。

```
*G0100
*
```

結果を確認しましょう。

Ｄ０４００ ↵

```
0400 : 20 21 22 23   !"#
(78)   24 25 26 27  $%&'
       28 29 2A 2B  ()*+
       2C 2D 2E 2F  ,-./
0410 : 30 31 32 33  0123
       34 35 36 37  4567
```

▼

０４００Ｈ～０４７ＦＨ番地に２０Ｈ～９ＦＨまでの16進数値が入っています。

```
0410 : 30 31 32 33  0123
(78)   34 35 36 37  4567
       38 39 3A 3B  89:;
       3C 3D 3E 3F  <=>?
0420 : 40 41 42 43  @ABC
       44 45 46 47  DEFG
```

４．ソースプログラムの作成・編集

本機のアセンブラでは、ＣＡＳＬのアセンブルと同様、ＴＥＸＴエリアに入れたソースプログラムをオブジェクトコードに変換（アセンブル）します。
変換したオブジェクトコードは、指定した番地のメモリ以降へ順番に入れていきます。

ここでは、ソースプログラムを作成するための仕様（入力形式など）について説明します。

４．１　ソースプログラムの構成

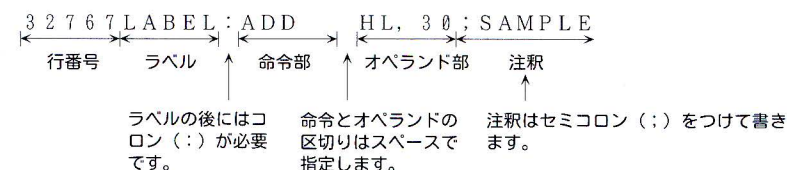
ソースプログラムの１行は通常、１ステートメント（１つの文）になります。
この行が何行か集まって、１つのプログラムが構成されます。
１つのソースプログラムはＯＲＧ命令から始まって、ＥＮＤ命令で終わります（ただし、この命令は省略可能です）。

```
〈例〉 10    ORG    0100H
          ≀
       100   END
```

- ＯＲＧ命令は、生成したオブジェクトを入れる番地を指定します。つまり、生成したオブジェクトは、ＯＲＧ命令で指定した番地から順番に入れられます。
- ＥＮＤ命令は、ソースプログラムの終わりを指定します。つまり、ＥＮＤ命令があると、そこでアセンブルを終了します。

これらの命令は、アセンブラを制御する命令ですので、オブジェクトには変換されません。（300ページの疑似命令参照）

◆行（ステートメント）の構成

１行は、行番号、ラベル、命令、オペランド、注釈、疑似命令などにより構成されます。

- １行の長さは、注釈を含めて最大254文字までです。
- アルファベットの小文字は、オペランドで文字列として書いた場合と、注釈に書いた場合以外は、大文字と同じものとみなします。

①行番号（ラインナンバー）

- 行番号は１～65279までの数値が使用できます。
- １～65279の範囲外の値を指定すると“ＬＩＮＥ　ＮＯ．ＥＲＲＯＲ”と表示されます。

②ラベル

- ラベルは行番号に続けて入力します。（ラベルと行番号との間にスペースを入れるとエラーになります。）
- ラベルは１～６文字まで使用できます。７文字以上入れるとエラーになります。
- ラベルには次の文字が使用できます。
  英文字（Ａ～Ｚ）、数字（０～９）、記号（［、］、＠、？、＿）。
  ただし、数字はラベルの先頭には使えません（行番号と区別ができなくなります）。
- 次のレジスタおよび条件コードと同じ文字（文字列）は単独で使えません。
  （シングルレジスタ）　Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｈ、Ｌ、Ｉ、Ｒ
  （ペアレジスタ）　ＡＦ、ＢＣ、ＤＥ、ＨＬ、ＩＸ、ＩＹ、ＳＰ
  （条件コード）　ＮＺ、Ｚ、ＮＣ、Ｃ、ＰＯ、ＰＥ、Ｐ、Ｍ
- ラベルの後ろにはコロン（：）が必要です。コロンがないとエラーになります。
  ただし、疑似命令のＥＱＵ命令で、ラベルに値を定義するときはコロンをつけません。
- ラベルを書かないときは、行番号と次の命令の間に１つ以上のスペース（空白）を入れてください。
  TAB を用いることもできます。

③命令部（オペコード）

- Ｚ80の命令をニモニックコードで書き表します。また、300ページ以降に記載している疑似命令も用います。なお、命令をオペコード（オペレーションコード）と呼びます。
- 命令部と、次のオペランドとの間は１つ以上のスペース（空白）で区切ります。
  TAB を用いることもできます。

④オペランド部

命令（オペコード）に対して、その実行対象となるレジスタ、アドレス（番地）、定数などをオペランドといいます。

- オペランドが２つ以上あるときは、コンマ（,）で区切ります。

● １つのオペランドは、32文字まで書くことができます。
● 定数には次のものが使用できます。

【数値定数】
２進数、10進数、16進数が使用できます。
２進数……０と１の数字で表し、最後にＢをつけます。
〈例〉　１０１１１１００Ｂ、１０００００Ｂ
10進数……０～９の数字で表します。
〈例〉　１８８、３２
16進数……０～９、Ａ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆの16進用数字で表し、最後にＨをつけます。数値がＡ、Ｂ、Ｃ、Ｄ、Ｅ、Ｆで始まる場合は、先頭に０をつけて他の命令と区別できるようにします。
〈例〉　０ＢＣＨ、２０Ｈ

【文字定数（文字列）】
文字列は’（シングルクォーテーション）で囲んで表します。文字のアスキーコードが定数になります。

〈例〉（文字列）	（指定）	（定数）
Ａ	’Ａ’	４１Ｈ
ＡＢ	’ＡＢ’	４１Ｈ、４２Ｈ
Ｂ’Ｃ	’Ｂ’’Ｃ’	４２Ｈ、２７Ｈ、４３Ｈ
’Ｄ	’’’Ｄ’	２７Ｈ、４４Ｈ
Ｅ’	’Ｅ’’’	４５Ｈ、２７Ｈ
’	’’’’	２７Ｈ
（ＮＵＬＬ）	’’	００Ｈ

【ラベル定数】
ＥＱＵ命令により、ラベルに定数を定義しておけば、そのラベルを定数として使用できます。
● 数式（四則計算）が使用できます。
次の符号と演算子が使用できます。ただし、計算の優先順位の判別は行いません。
（符　号）　＋、－
（演算子）　＊、／、＋、－
● 計算は16ビットで行われます。オーバーフロー（桁あふれ）が起こっても、オーバーフロー分は無視します（エラーにはなりません）。計算結果の８ビットまたは16ビットを使用してオブジェクトを生成します。
● 数式が書けるところでは、バイト、ワードのチェックは行いません。
〈例〉　ＬＤ　Ａ，４１４２Ｈ　→　ＬＤ　Ａ，４２Ｈ　とみなされます。
　　　ＤＢ　１２３４Ｈ　→　ＤＢ　３４Ｈ　とみなされます。

⑤注釈（コメント）
各行はセミコロン（；）をつけて注釈を書くことができます。セミコロン以降は、行の終わり（行末）まで注釈とみなされ、機械語（オブジェクト）には変換されません。

４．２　ソースプログラムの消去
ＴＥＸＴモードの機能選択画面で、Ⓓを押せばデリート（Del）機能が選ばれ、次のようにテキスト内容を消去（削除）してよいか聞いてきます。（テキスト内容がない場合は、画面は変わりません。）

ＴＥＸＴ　ＤＥＬＥＴＥ　ＯＫ？　（Ｙ）

Ⓨを押せば、テキスト内容がすべて消去され、ＴＥＸＴモードの機能選択画面に戻ります。
くわしくは174ページを参照してください。

４．３　ソースプログラムの入力
ＴＥＸＴモードの機能選択画面でⒺを押せばエディット機能が選ばれます。

```
TEXT EDITOR
<
```

次にソースプログラムの入力手順を説明します。
①行番号を入力します。
②ラベルがないときは、[TAB]を押します。カーソルが命令部に移ります。
[TAB]の代わりに[SPACE]でスペースを入力してもかまいません。
スペースは１つ以上入れてください。
ラベルを入力するときは、行番号に続けて入力し、コロン（：）を入れます。この後ろはスペースを入れても、入れなくてもかまいません。
③命令を入力します。
命令の後にオペランドがある場合は、[TAB]または[SPACE]でスペースを１つ以上入れます。
④オペランドを入力します。
オペランドが２つ以上ある場合は、コンマ（，）で区切って入力します。
⑤注釈をつける場合は、セミコロン（；）を入力して、その後に入力します。
⑥１行の入力を完了したら[↵]を押して、プログラムをメモリに格納します。[↵]を押せばカーソルが消えます。
次の行を入力するときは、①から繰り返します。
289ページのソースプログラムの入力例を参照してください。

５．アセンブル
ＴＥＸＴ（テキスト）モードで入力したソースプログラムをアセンブル（オブジェクトに変換）します。
288ページのプログラム例が入力されているものとして説明します。プログラムを入力しておいてください。

５．１　メニュー画面
アセンブルする場合、まずアセンブラモードにします。

[▼] を押せば次へ進みます。

エラー内容（定義されていないラベルを使っている）

[▼] で次へ進みます。

```
object：0100H－010CH
size  ：000DH(    13)bytes
label ：   2
error ：   2
```

アセンブルの終了画面になります。

このときは、“complete！”は表示されません。

この後、[CLS] を押すと、アセンブラのメニュー画面に戻ります。

- この例では、50行の命令はないものとみなされ、80行のラベルは0000H番地の指定とみなされて、アセンブルされます。
- エラーが検出された場合、生成されたオブジェクトプログラムは、正しいプログラムにはなりません。これを実行すると暴走したり、記憶内容を壊したりする場合があります。ソースプログラムを修正して、再度アセンブルしてから、エラーのないプログラムを実行してください。

5.3　アセンブルリストを表示させる方法

ソースプログラムをアセンブルした場合に、各行で生成されるオブジェクトおよび格納されるアドレスなどを確認することができます。

アセンブラのメニュー画面で、[D] を押せばアセンブルリストの表示が開始されます。

以降 [▼] で表示を送っていきます。

288ページのプログラム例が入っているものとして、実行してみましょう。

メニュー画面で [D] を押します。

オブジェクト欄が空白の場合は、オブジェクトがありません。8桁を超えるときは8桁ごとに行を変えて表示します。

以降 [▼] で表示を送って確認していきます。

```
                LD      HL, 04
      00H
0105 77                     40
      LBL：     LD      (HL),
      A
0106 3C                     50
                INC     A
0107 23                     60
                INC     HL
0108 FEA0                   70
                CP      0A0H
010A C20501                 80
                JP      NZ, LB
      L
010D C9                     90
                RET
010E                       100
              END

 ＊＊＊＊ SYMBOL TABLE ＊＊＊＊
START ：0100 LBL    ：0105
object：0100H－010DH
size  ：000EH(    14)bytes
label ：   2
error ：   0     complete ！
```

シンボルテーブル※
オブジェクト格納番地
オブジェクトの大きさ
ラベル数
エラー件数

※シンボルテーブルは、ラベルに与えられる値を16進数で示します。

- リスト表示中に [CLS] を押せば、メニュー画面に戻ります。
- リスト表示では、生成したオブジェクトを機械語エリアに格納しません。格納するには、アセンブルを行ってください。

5.4　アセンブルリストを印字させる方法

CE-126P（プリンタ）をお持ちの場合は、本機に、プリンタCE-126Pを接続してプリンタの電源を入れ、アセンブラのメニュー画面で [P] を押すと、アセンブルリストが印字されます。

- プリンタの電源が入っていない場合や接続されていない場合などには、[P] を押したとき“＊PRINTER ERROR”と表示されます。このときは [CLS] でエラーを解除し、プリンタを確認してください。
- 画面右下の“PRINT”が点灯していなくても、リストは印字できます。
  リストの印字が終了するとアセンブル終了画面になります。[CLS] を押せばメニュー画面に戻ります。

印字の様式はリスト表示の様式と同じです。

〈印字例〉

```
 **** ASSEMBLE LIST ****
0100                    10
             ORG    0100H
0100 3E20               20
       START:LD     A,20H
0102 210004             30
             LD     HL,04
       00H
0105 77                 40
       LBL:  LD     (HL),
       A
0106 3C                 50
             INC    A
0107 23                 60
             INC    HL
0108 FEA0               70
             CP     0A0H
010A C20501             80
             JP     NZ,LB
       L
010D C9                 90
             RET
010E                   100
            END

 **** SYMBOL TABLE ****
START :0100 LBL   :0105
object:0100H-010DH
size  :000EH(    14)bytes
label :    2
error :    0   complete !
```

● リストの印字を途中で中止させる場合は、印字が止まるまで BREAK を押してください。
印字を中止すると“－－－ break －－－”と表示されます。CLS を押せば、メニュー画面に戻ります。

アセンブラのメニュー画面で L を押すと、ミニＩ／Ｏのパラレルポートからアセンブルリストが送出されます。詳しくは、先生の指導に従ってください。（366ページを参照）

６．アセンブラの疑似命令の説明

疑似命令は、アセンブラを制御する命令で、それ自体はオブジェクトに変換されません。
本機のアセンブラには、次の疑似命令があります。

● 格納開始番地の指定
　ＯＲＧ命令
● データの定義
　ＤＥＦＢ／ＤＢ／ＤＥＦＭ／ＤＭ命令
　ＤＥＦＳ／ＤＳ命令
　ＤＥＦＷ／ＤＷ命令
● ラベルの値の定義
　ＥＱＵ命令
● アセンブル終了の指定
　ＥＮＤ命令

次に、これらの疑似命令の機能を説明します。

〈書式で使用している用語、記号の意味〉

式	数値、数式、ラベル、’文字列’を書くことができます。
数式	数値、ラベル、およびこれらを用いた四則計算式を書くことができます。
｛ ｝	｛ ｝内に併記されている内容のいずれか１つを選ぶことを示します。
［ ］	［ ］で囲まれた部分の省略ができることを示します。
［ ］…	［ ］で囲まれた部分を省略、または繰り返しができることを示します。

ＯＲＧ……オリジン

機　能　オブジェクトの格納開始番地（アドレス）および、実行番地を指定します。

書　式　ＯＲＧ［式$_1$，］式$_2$

説　明
● 式$_2$により、アセンブルをして生成されたオブジェクトの格納開始番地を指定します。
式$_2$で指定した番地以降に、オブジェクトが順番に格納されます。
● 式$_1$により、最終的にオブジェクトプログラムを格納し、実行する場所の開始番地を指定します。（次ページの ご注意 を参照）
● 式$_1$を省略した場合は、式$_2$と同じ番地を指定したものとみなされます。
● ソースプログラムにＯＲＧ命令を書かなかった場合は、ＯＲＧ　１００Ｈ　の設定があるものとみなして、オブジェクトの生成、格納が行われます。
● 式$_1$は０ＨからＦＦＦＦＨの範囲内で指定できます。ただし、オブジェクトプログラムがこの範囲を超えないように指定してください。（この範囲でも、メモリの制約によりエラーになる場合があります。）

● 式$_2$は９０Ｈから、機械語モニタのＵＳＥＲ命令で確保した機械語エリアの範囲内で指定できます。ただし、オブジェクトプログラムがこの範囲を超えないように指定してください。

ご注意

先の説明のように、アセンブルして生成したオブジェクトは式$_2$で指定した番地から格納されます。そのまま（この場所で）オブジェクトプログラムを実行する場合、式$_1$は式$_2$と同じ番地にします。（このとき、式$_1$の指定は省略できます。）

図のように、生成したオブジェクトプログラムをパソコン等に記録し、それを別の場所に読み込んで実行する場合は、読み込む場所の番地を式$_1$で指定します。

式$_1$と式$_2$で別々のアドレスを指定した場合は、アセンブル実行後、必ず上図のようにオブジェクトプログラムを式$_1$で指定した番地で始まるメモリに移し変えて、実行してください。
そのまま実行すると、プログラムによっては暴走する恐れがあります。

● 式$_1$と式$_2$に別々の番地が指定されると、アセンブラは、式$_1$で指定された番地からオブジェクトを格納するものとみなしてアセンブルします（実際は、式$_2$で指定した番地以降に格納します。）
つまり、ソースプログラムのラベル等に与えられる番地は、式$_1$で指定された番地を基準にした値になります。
このため、生成されたオブジェクトが格納された番地と、プログラムに与えられた番地が異なります。
したがって、このオブジェクトプログラムを式$_1$で指定した番地に移し変えないで実行すると、暴走する可能性があります。

例　ＯＲＧ　０４００Ｈ
　　０４００Ｈ番地からオブジェクトを格納します。
　ＯＲＧ　０Ｃ００Ｈ，０１００Ｈ
　　０Ｃ００Ｈ番地を開始番地としてアセンブルし、生成したオブジェクトを０１００Ｈ番地以降に格納します。

ＤＥＦＢ／ＤＢ／ＤＥＦＭ／ＤＭ……デファイン　バイト／デファイン　メッセージ

機　能　オペランドに記述した数値の下位１バイト、または文字列をオブジェクトに生成します。

書　式

［ラベル：］ {ＤＥＦＢ / ＤＢ / ＤＥＦＭ / ＤＭ} 式［，式］…

説　明　● オペランドに記述された数値の下位１バイトがオブジェクトに生成されます。
〈例〉　ＤＥＦＢ　１２３４ｨ　　３４Ｈがオブジェクトになります。
　　　ＤＢ　１２３４　　Ｄ２Ｈがオブジェクトになります。（10進数１２３４は16進数では４Ｄ２です。）

● 文字列は、’（シングルクォーテーション）で囲んで記述します。32文字まで記述できます。
文字列は、その１文字１文字のアスキーコードがオブジェクトとして生成されます。
〈例〉　ＤＥＦＭ　’ＤＡＴＡ’　　４４Ｈ、４１Ｈ、５４Ｈ、４１Ｈがオブジェクトになります。

● オペランドが複数ある場合は、，（コンマ）で区切って記述します。
〈例〉　ＤＢ　３２＊４＋５，’Ｘ２’　　８５Ｈ、５８Ｈ、３２Ｈがオブジェクトになります。

例

〈ソースプログラム〉			〈オブジェクトプログラム〉
１０	ＯＲＧ	０１００Ｈ	
２０	ＬＤ	ＨＬ，ＤＡＴＡ	２１　０Ｃ　０１
３０	ＬＤ	ＤＥ，３００Ｈ	１１　００　０３
４０	ＬＤ	ＢＣ，５	０１　０５　００
５０	ＬＤＩＲ		ＥＤ　Ｂ０
６０	ＲＥＴ		Ｃ９
７０ＤＡＴＡ：	ＤＢ	’ＡＢＣＤＥＦＧＨ’	４１　４２　４３　４４ ４５　４６　４７　４８
８０	ＥＮＤ		

アドレス	内容
0100	
0101	
：	
010C	41
010D	42
010E	43
010F	44
0110	45
0111	46
0112	47
0113	48
0114	
：	
0300	41
0301	42
0302	43
0303	44
0304	45
0305	

70行の文字列は、１字１字がオブジェクトに変換されます。
このプログラムは、ラベルＤＡＴＡで示す番地以降にあるデータを５バイト分だけ、３００Ｈ番地以降へコピーします。
つまり、４１Ｈ、４２Ｈ、４３Ｈ、４４Ｈ、４５Ｈを３００Ｈ～３０４Ｈ番地にコピーします。

ＬＤＩＲ命令について

ＬＤＩＲだけで次の一連の実行がなされます。くわしくはＺ80に関する市販の書籍を参照してください。

（ＤＥ）←（ＨＬ）	ＨＬで指定されたアドレスの内容をＤＥで指定されたアドレスに格納します。 アドレスは間接指定で、最初は０１０Ｃのアドレスの内容を３００のアドレスに格納します。
ＤＥ←ＤＥ＋１	ＤＥ、ＨＬレジスタの値をカウントアップします。
ＨＬ←ＨＬ＋１	
ＢＣ←ＢＣ－１	ＢＣレジスタの値をカウントダウンします。
Repeat until ＢＣ＝０	ＢＣレジスタの値が０になるまで繰り返します。

エラーの種類・エラーメッセージ		内　容（原因）
フォーマット　エラー	FORMAT ERROR (1)	オペランドの区切り子に誤りがある。
	FORMAT ERROR (2)	数値などに特殊なコード（アスキーコードの01H～1FHなど）や文字が入っている。（ただし、このようなコードや文字は、通常の入力操作では入りません。）
	FORMAT ERROR (3)	オペランドの数に誤りがある。
	FORMAT ERROR (4)	ラベルに、使えない文字を使用している。
	FORMAT ERROR (5)	ラベルの文字数が6文字を超えている。
	FORMAT ERROR (6)	文字列を指定するシングルクォーテーションが閉じていない。
	FORMAT ERROR (7)	命令や、個々のオペランドなどの、文字数が32文字を超えている。（たとえば、オペランドの数値（番地など）の前に、不要な0を多くつけた場合など）
アンデファインド　シンボル	UNDEFINED SYMBOL	定義されていないシンボル（ラベル）を使っている。
マルチ　デファイン　シンボル	MULTI DEFINE SYMBOL	同じシンボル（ラベル）が2回以上定義されている。
ファイル　ノット　エグジスト	FILE NOT EXIST	ＴＥＸＴエリアにアセンブルするプログラムが入っていない（存在しない）。
ユーザーエリア　オーバー	USER AREA OVER	機械語エリア内にオブジェクトが格納できない。（ＯＲＧ命令で指定した格納開始番地が機械語エリアの外になっている。または、オブジェクトを格納していくと、途中で機械語エリアをはみ出してしまう。）
ワークエリア　オーバー	WORK AREA OVER	フリーエリア（未使用のエリア）が小さく、アセンブル用のワークエリアが確保できない。（他の機能からアセンブラ機能に移ったときや、アセンブル中に、ワークエリアが確保できない。）
プリンタ　エラー	PRINTER ERROR	プリンタに異常がある（プリンタが接続されていない、プリンタの電源が入っていない、プリンタの電池の消耗などでプリンタが動作しない）。

第9章　ＰＩＣ

1．ＰＩＣ

PIC（Peripheral Interface Controller）とは、コンピュータの周辺に接続される周辺機器の接続部分を制御するために米マイクロチップテクノロジー社が開発した“マイコン”と呼ばれる領域のＩＣです。
PICについては、入門書等の関連書籍が市販されていますので、それらの書籍を参照してください。
この章では、ＰＩＣ用プログラムの作成方法について説明します。
なお、本機でプログラムの作成が可能なＰＩＣ（14ビット・コアFlashメモリのＰＩＣ）は次のとおりです。

2001年7月9日現在

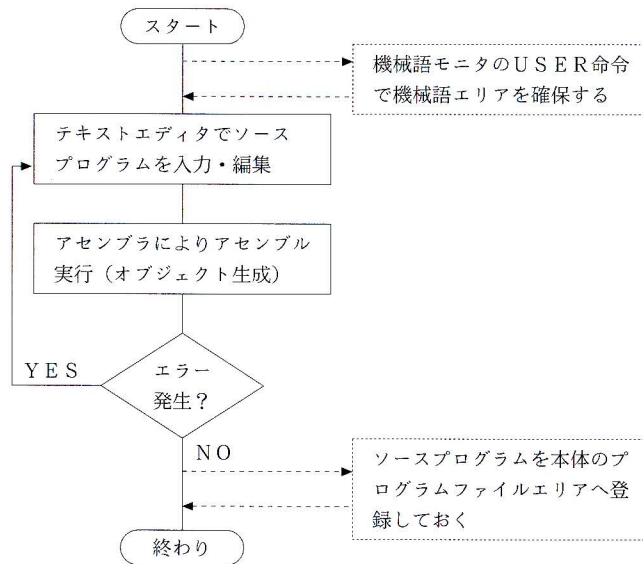
品　名	プログラムメモリ	ピン数
PIC16F627	1Ｋワード	18ピン
PIC16F84	1Ｋワード	18ピン
PIC16F84A	1Ｋワード	18ピン

ＰＩＣ製品名や開発ツールなどの固有名詞は米国マイクロチップ・テクノロジー社の商品名などです。

2．ＰＩＣプログラムの作成手順

ＰＩＣプログラムの作成手順とその準備について説明します。
なお、ＰＩＣプログラムの詳しい使いかたについては、先生の指導に従ってください。

２．１　プログラム作成手順

ＰＩＣのプログラムの作成は次の手順で行います。

【作成手順の解説】

①モニタ機能を使用して機械語エリアを確保します。
②ＴＥＸＴエディタ機能を使用してソースプログラムを入力します。
③ＰＩＣモードのアセンブラ機能によりオブジェクトプログラムに変換します。
④必要に応じて②〜③を繰り返します。
⑤必要であればソースプログラムの本体のプログラムファイルエリアへの登録、印字を行います。

２．２　機械語エリア（ユーザーエリア）の確保

機械語エリアは機械語モニタのＵＳＥＲ命令で確保します。（275ページ参照）
確保しておかないとソースプログラムの変換などができません。
機械語モニタモードにします。

BASIC MON ↵

```
MACHINE LANGUAGE MONITOR
*
```

ＵＳＥＲ命令で機械語エリアを確保します。
本機では、１Ｋワード（３ＫＢ）までのＰＩＣ用プログラムを作成することができます。
したがって、０ＣＦＦＨまで（０１００Ｈ〜０ＣＦＦＨ）を確保しておきます。

ＵＳＥＲ０ＣＦＦ ↵

```
MACHINE LANGUAGE MONITOR
*USER0CFF
FREE:0100-0CFF
*
```

確保した機械語エリアが表示されます。

３．ソースプログラムの作成・編集

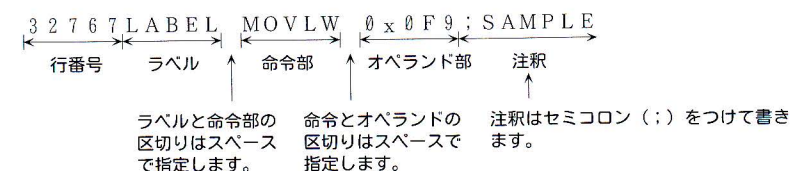
本機のアセンブラでは、ＣＡＳＬのアセンブルと同様、ＴＥＸＴエリアに入れたソースプログラムをオブジェクトコードに変換します。（マイクロチップ・テクノロジー社のパソコン用開発ツール：ＭＰＬＡＢに準拠）

ここでは、ソースプログラムを作成するための仕様（入力形式など）について説明します。

３．１　ソースプログラムの構成

ソースプログラムの１行は通常、１ステートメント（１つの文）になります。
この行が何行か集まって、１つのプログラムが構成されます。

◆行（ステートメント）の構成

１行は、行番号、ラベル、命令、オペランド、注釈、疑似命令などにより構成されます。

● １行の長さは、注釈を含めて最大254文字までです。
● アルファベットの小文字は、オペランドで文字定数として書いた場合と、注釈に書いた場合以外は、大文字と同じものとみなします。

①行番号（ラインナンバー）

● 行番号は1～65279までの数値が使用できます。

● 1～65279の範囲外の値を指定すると“LINE NO. ERROR”と表示されます。

②ラベル

● ラベルは行番号に続けて入力します。（ラベルと行番号との間にスペースを入れると命令とみなされます。）

● ラベルは1～8文字まで使用できます。9文字以上入れるとエラーになります。

● ラベルには次の文字が使用できます。
英文字（A～Z）、数字（0～9）、記号（_）。
ただし、数字はラベルの先頭には使えません（行番号と区別ができなくなります）。

● ラベルと、次の命令部の間は1つ以上のスペース（空白）で区切ります。TAB を用いることもできます。

● ラベルを書かないときは、行番号と次の命令の間に1つ以上のスペース（空白）を入れてください。
TAB を用いることもできます。

● ラベルは最大102個まで定義できます。

③命令部（オペコード）

● 14ビット・コア用の35命令をニモニックコードで書き表します。また、311ページ以降に記載している疑似命令、#INCLUDE命令も用います。なお、命令をオペコード（オペレーションコード）と呼びます。

● 命令部と、次のオペランドとの間は1つ以上のスペース（空白）で区切ります。
TAB を用いることもできます。

④オペランド部

命令（オペコード）に対して、その実行対象となるレジスタ、アドレス（番地）、定数などをオペランドといいます。

● オペランドが複数あるときは、コンマ（,）で区切ります。

● 定数には次のものが使用できます。

【数値定数】
10進数、16進数が使用できます。

10進数……0～9の数字で表します。
〈例〉 188、32
16進数……先頭に0xをつけ、0～9、A、B、C、D、E、Fの16進用数字で表します。
〈例〉 0xBC、0x20

【文字定数】
文字は’（シングルクォーテーション）で囲んで表します。文字のアスキーコードが定数になります。

〈例〉	（文字）	（指定）	（定数）
	A	’A’	0x41
	（NULL）	’’	0x0

【ラベル定数】
EQU命令により、ラベルに定数を定義しておけば、そのラベルを定数として使用できます。

⑤注釈（コメント）

各行はセミコロン（;）をつけて注釈を書くことができます。セミコロン以降は、行の終わり（行末）まで注釈とみなされ、機械語（オブジェクト）には変換されません。

3.2 ソースプログラムの消去

TEXTモードの機能選択画面で、D を押せばデリート（Del）機能が選ばれ、次のようにテキスト内容を消去（削除）してよいか聞いてきます。（テキスト内容がない場合は、画面は変わりません。）

TEXT DELETE OK? (Y)

Y を押せば、テキスト内容がすべて消去され、TEXTモードの機能選択画面に戻ります。
くわしくは174ページを参照してください。

3.3 ソースプログラムの入力

TEXTモードの機能選択画面で E を押せばエディット機能が選ばれます。

```
TEXT EDITOR
<
```

次にソースプログラムの入力手順を説明します。

①行番号を入力します。
②ラベルがないときは、TAB を押します。カーソルが命令部に移ります。
TAB の代わりに SPACE でスペースを入力してもかまいません。
スペースは1つ以上入れてください。
ラベルを入力するときは、行番号に続けて入力します。この後ろは、TAB または SPACE でスペースを1つ以上入れます。
③命令を入力します。
命令の後にオペランドがある場合は、TAB または SPACE でスペースを1つ以上入れます。
④オペランドを入力します。
オペランドが複数ある場合は、コンマ（,）で区切って入力します。
⑤注釈をつける場合は、セミコロン（;）を入力して、その後に入力します。
⑥1行の入力を完了したら ↵ を押して、プログラムをメモリに格納します。↵ を押せばカーソルが消えます。
次の行を入力するときは、①から繰り返します。

4．アセンブラ

ＴＥＸＴモードで入力したソースプログラムをＰＩＣマイコンが実行できるオブジェクトプログラムに変換する場合、まず、ＰＩＣモードにします。

SHIFT ＋ ASMBL を押してから P を押します。

メニュー画面になります。

A を押します。

アセンブルが実行されます。

アセンブル実行中は画面の下の行に“Assembling...”と表示され、終われば“Complete! (***** words)”と表示されます。アセンブルするプログラムが短いときは、この表示は一瞬で終わります。（*****は、データ変換サイズ（word単位）を示します。）

4．1　アセンブラの疑似命令

疑似命令は、アセンブラを制御する命令で、それ自体はオブジェクトに変換されません。

本機のアセンブラには、次の疑似命令があります。

- コンフィグレーションビットの指定
  ＿＿ＣＯＮＦＩＧ命令
- プログラム開始番地の指定
  ＯＲＧ命令
- ラベルの値の定義
  ＥＱＵ命令
- データの定義
  ＤＷ命令

次に、これらの疑似命令の機能を説明します。

〈書式で使用している用語、記号の意味〉

式　　数値、ラベル、’文字’を書くことができます。

［ ］　　［ ］で囲まれた部分の省略ができることを示します。

＿＿ＣＯＮＦＩＧ……コンフィグ

機　能　コンフィグレーションビットの設定をします。

書　式　＿＿ＣＯＮＦＩＧ　式

説　明　●各ＰＩＣに従ったコンフィグレーションビットの設定をします。

コンフィグレーションビットの詳細は各ＰＩＣの説明書を参照してください。

●ＭＰＡＳＭでは“&”（ビットのＡＮＤ）による指定ができますが、本機ではできません。

ＯＲＧ……オリジン

機　能　プログラムの開始番地（アドレス）を指定します。

書　式　ＯＲＧ　式

説　明　●ソースプログラムにＯＲＧ命令を書かなかった場合は、ＯＲＧ　０　の設定があるものとみなして、０番地からプログラムが開始されます。

●式$_1$は０Ｈから１ＦＦＦＨの範囲内で指定できます。ただし、オブジェクトプログラムがこの範囲を超えないように指定してください。（この範囲でも、メモリの制約によりエラーになる場合があります。）

〈例〉　ＯＲＧ　０ｘ０００６

ＥＱＵ……イー・キュー・ユー

機　能　ラベルにオペランドで指定した値を与えます。

書　式　ラベル　ＥＱＵ　式

説　明　●ラベルに、オペランドで指定した値を定義付けます。式の演算はできません。

●１または２バイトの値（数値または’文字’）を指定します。

〈例〉　ＳＴＡＲＴ　ＥＱＵ　０ｘ１０００　ラベルＳＴＡＲＴに、０ｘ１０００が定義付けられます。ＳＴＡＲＴは定数０ｘ１０００として扱うことができます。

ＯＫ　ＥＱＵ　’Ｙ’　ラベルＯＫに、０ｘ５９が定義付けられます。

ＤＷ……デファイン　ワード

機　能　オペランドに記述した数値の下位２バイトをオブジェクトに生成します。

書　式　［ラベル：］　ＤＷ　式

説　明　●オペランドに記述された数値の下位２バイトをオブジェクトに生成します。

ただし、３ＦＦＦＨ（14ビット）を超えた場合は14ビットでカットされます。

〈例〉　ＤＷ　０ｘ１２３４　０ｘ１２３４がオブジェクトになります。

4．2　＃ＩＮＣＬＵＤＥ

機　能　指定されたファイルをソースコードの一部として読み込み、標準的なラベルを使うことが可能になります。

書　式　＃ＩＮＣＬＵＤＥ　”ファイル名”

説　明　●ＭＰＡＳＭにある同名ファイルのラベル定義を、本機に保有しており使用できます。

ファイル名は”（ダブルクォーテーション）で囲んで表します。

なお、指定できるファイル名は次のとおりです。

P16F627.INC、P16F83.INC、P16F84.INC、P16F84A.INC

…14ビット・コアFlashメモリＰＩＣ

および

PIC．INC…上記以外

- 定義できるラベル数102個には含まれません。
- プログラムの先頭で指定します。
- MPASMで定義されている8文字を超えるラベルは、本機では次のとおり指定してください。

MPASMで定義されているラベル名	本機での指定
OPTION_REG	OPTION_R
NOT_T1SYNC	NOT_T1SY

4.3　エラーメッセージ

アセンブラ機能を使用するときに、発生することがあるエラーのエラーメッセージと、その内容を記載します。エラーは CLS で解除できます。また、BASIC や TEXT などでモードを切り替えた場合も解除されます。

エラーメッセージ	内　容（原　因）
File not exist!	TEXTエリアにアセンブルするプログラムが入っていない（存在しない）。
No USER AREA!	機械語エリアが確保されていない。
	ラベルエリア分の機械語エリアが確保されていない。
Not __CONFIG data!	__CONFIG疑似命令がない。
Syntax error!　(*****)	__CONFIGの前にラベルがある。
	ORG指定の前にラベルがある。
	EQU指定の前にラベルがない。
	不当ニモニック、疑似命令がある。
	ニモニック、疑似命令のオペランドの後ろに、SPACE/TAB/CR/;以外がある。
	オペランドなしの命令の後ろに、SPACE/TAB/CR/;以外がある。
	オペランド1と2の間に、SPACE/TAB/,以外がある。
	オペランド1と2の間に、，がない。
	不当なオペランドがある。
	不当なプリプロセッサコマンドがある。
	プリプロセッサコマンドのフォーマットに間違いがある。
Out of range!　(*****)	結果格納先指名子が1ビット以上ある。
	リテラル値が8ビット以上ある。
	ビット指定値が3ビット以上ある。
	オペランド値等が16ビット以上ある。
Undefined label!　(*****)	オペランドで使用しているラベルがない。
Undefined line!　(*****)	アドレス値が機械語エリアを超えている。
Label too long!　(*****)	ラベルが8文字以上ある。
Out of memory!　(*****)	ORG指定が8K以上を指している。
	オペコードが機械語エリアを超えている。
	ラベル数がMaxを超えている。

エラーメッセージ	内　容（原　因）
Multi define!　(*****)	同一ラベルが複数ある。
	#includeが複数ある。
'Not include file!　(*****)	指定includeファイル名が不当。

（***** は、エラー発生行番号を示します。）

第10章
ＢＡＳＩＣの各命令の説明

この章ではＢＡＳＩＣの各命令を１つ１つ説明します。
本機がどのような命令をもっているかを見て頂いた後で、必要な命令の説明をお読みください。

以降の説明で書式などに使用する用語の意味を示します。
式……………数値、数値変数およびこれらを含んだ計算式を示します。
変数…………配列要素を含んだ数値変数、文字変数を示します。
〃文字〃 ……〃〃（ダブルクォーテーションマーク）で囲まれたキャラクタ（文字、数字、記号）を示します。
文字列………〃文字〃、文字変数を示します。
（ ）………カッコでくくる必要があることを示します。
［ ］………省略可能であることを示します。ただし、この後にほかの指定を行う場合は、コンマなどの区切り記号が必要です。
{A / B}………ＡあるいはＢを選択することができます。
[プログラム]…プログラムによる実行が可能です。
[マニュアル]…マニュアル操作による実行が可能です。
省略形………命令の中には、そのつづりを省略した形で入力できるものがあり、その場合は最も省略した形で示しています。

〈例〉 省略形…Ｐ． これはＰＲＩＮＴの省略形ですが、次の形でも有効になります。
ＰＲ．
ＰＲＩ．
ＰＲＩＮ．

（注）・変数などに続いてＢＡＳＩＣ命令を入力する場合は、変数と命令の間にスペースを入力してください。

〈例〉 ５０ ＩＦ Ａ＝Ｂ ＴＨＥＮ １００
（↑ スペースを入れる。）

・関数命令などをマニュアルで実行する場合はＲＵＮモードで実行してください。
ＰＲＯモードで実行するとエラー12になります。
ただし、ＰＲＩＮＴ命令の中で使用すれば、ＰＲＯモードでも使用できます。

〈例〉 ＰＲＩＮＴ ＣＨＲ＄ ９０[↵] → Ｚ

関数

ＡＮＤ…………アンド
省略形……ＡＮ．
[プログラム] [マニュアル]

[機能] 式と式との論理積を計算します。また、条件式の結合を行います。

[書式] 式 ＡＮＤ 式
条件式 ＡＮＤ 条件式

[参照] ＯＲ、ＮＯＴ、ＩＦ

[説明]
● ２進数において、論理積は次のような値を取ります。

１ ＡＮＤ １＝１ 　 ０ ＡＮＤ １＝０
１ ＡＮＤ ０＝０ 　 ０ ＡＮＤ ０＝０

10進数の論理積を求めた場合は、計算機内では10進数を２進数に変換してから各桁の論理積を求め、その結果を10進数に戻します。
たとえば、41と27の論理積は次のように計算されます。

```
４１ ＡＮＤ ２７＝９
      101001………41
ＡＮＤ〈
      011011………27
      ─────────
      001001……… 9
```

41と27をそれぞれ２進数に変換し、各桁のＡＮＤを取ります。そして、その結果を10進数に変換すれば９になります。

● 式の値は―32768～32767の整数部が有効になります。
● ２つ以上の条件をすべて満足するような条件を１つの式で表します。

〈例〉 ５０ ＩＦ Ｂ＞５ ＡＮＤ Ｃ＞＝４ ＴＨＥＮ………
もし、Ｂが５よりも大きく、かつＣが４よりも大きいか等しいとき、ＴＨＥＮに続く命令を実行します。

関数

ＡＳＣ…………アスキー
省略形……ＡＳ．
[プログラム] [マニュアル]

[機能] 文字や記号、数字などをキャラクタコードに変換します。

[書式] ＡＳＣ {〃文字〃 / 文字変数}

[参照] ＣＨＲ＄

[説明]
● 文字や記号、数字などをキャラクタコードに変換します。たとえば「Ｚ」という文字のキャラクタコードを知りたい場合は

Ａ＝ＡＳＣ〃Ｚ〃

として実行すれば、Ａには「Ｚ」のキャラクタコードが10進数の90として代入されます。
文字が２文字以上指定された場合は、先頭の文字のみがキャラクタコードに変換されます。

〈例〉 １０ ＣＬＳ
２０ Ａ＄＝ＩＮＫＥＹ＄

```
30 IF A$=""THEN 20
40 B=ASC A$
50 PRINT A$;"␣␣";B
```

プログラムをスタートさせた後、アルファベットや数字キーなどを押せば、その文字のキャラクタコードを表示します。

● 文字、数字、記号など（これらを総称してキャラクタといいます）を計算機が記憶したり、処理したりする場合は、すべて計算機が取り扱いやすい数値に変換します。たとえばアルファベットのAは計算機内では65（10進数）という数値（コード）になっています。（実際には2進数の01000001となっています。）同様にBは66、Cは67というようにコードを決めています。このコードの決めかたに何種類かあって、代表的なものにアスキーコード（ASCII code）とJISコードがあります。
本機ではJISコードを元にして作成されたキャラクタとそのコードを使用しています。（384ページのキャラクタ・コード表を参照してください。）

基本命令

AUTO………オート
省略形……なし　［マニュアル］

［機能］　行番号の自動発生を行います。（PROモードのマニュアル操作でのみ有効）

［書式］　AUTO［[開始行]［，増分]］↵

［参照］　NEW、PASS

［説明］
- 最初に開始行で指定された行番号を発生して、入力待ちになります。1行分のプログラムを入力して↵を押すと、増分で指定した値を加えた行番号を発生して入力待ちになります。
以下、同様に↵を押すたびに新たな行番号を発生します。
- 開始行や増分に0、負数、65279を超える値を指定したときはエラーになります。
- 開始行や増分を省略したときは、それぞれ10が指定されます。
ただし、前回AUTO命令で指定された値があれば、その値が指定されます。
- PROモードからRUNモードに移り、再びPROモードに戻ったとき、AUTO↵と操作すると、前の設定値が有効で、引き続いた行番号を発生します。
また、行番号を表示しているときに［CLS］や［ON］(BREAK)を押したときも同じです。
- パスワードが設定されているときは、AUTO命令は無視されます。
- 次の場合は設定値（開始行、増分値）が解除されます。
  - ［SHIFT］＋［CLS］(CA)と操作したとき
  - 電源をオフ、オンしたとき
  - LOAD、NEW命令を実行したとき
  - BLOAD命令を実行し、BASICプログラムが読み込まれたとき
  - FILES命令により一覧されたファイルから選択後、［SHIFT］＋［M］(LOAD)の操作をしたとき
  - BASICコンバータでBASICへの変換（Basic←Text）を実行したとき

〈例〉　AUTO 100，10 ↵
開始行番号を100とし、110、120、……と行番号を発生します。

ポケコン間通信命令

BLOAD……バイナリ・ロード
省略形……BLO.　［マニュアル］

［機能］　別のポケコンから本機にBASICプログラムを読み込ませます。（PROおよびRUNモードのマニュアル操作でのみ有効）

［書式］　BLOAD↵

［参照］　BLOAD?、BSAVE

［説明］
- 別のポケコンに記録されているBASICプログラムを本機に読み込みます。
- 読み込みが行われているときは、画面右下に＊マークが表示されます。読み込みが終われば、＊マークは消えます。
BASICプログラムの受信待ちのときは、まだ読み込みが行われていませんので＊マークは表示されません。
これは、BLOAD?命令でも同じです。
なお、読み込みが終われば、プロンプト記号“>”が表示されます。
- 別のポケコンにBASICプログラムがなかった場合でも、本機はBASICプログラムを検索し続けます。この場合は［ON］(BREAK)を押して検索を止めてください。これは、BLOAD?命令でも同じです。
- 実行中にエラーが発生すると、計算機内のBASICプログラムが無効になります。
- 実行するとオープンされていた全ファイルが閉じられます。（クローズされます。）

ポケコン間通信命令

BLOAD　M…バイナリ・ロード・エム
省略形……BLO.　M　［マニュアル］

［機能］　別のポケコンから機械語プログラムを読み込みます。

［書式］　BLOAD　M［ロード番地］↵

［参照］　BLOAD、BSAVE　M

［説明］
- BSAVE　M命令で別のポケコンに書き込んだ機械語プログラムを読み込みます。そのとき、本体のプログラムは消去しません。
- 「ロード番地」を指定したときは、指定した番地を開始番地として読み込みます。省略すると、BSAVE　M命令で指定した番地に読み込みます。

ポケコン間通信命令

BLOAD?…バイナリ・ロード?
省略形……BLO.　?　［マニュアル］

［機能］　本機内のBASICプログラムと別のポケコンに記録されている内容との照合を行います。（PROおよびRUNモードのマニュアル操作でのみ有効）

［書式］　BLOAD?↵

［参照］　BLOAD、BSAVE

［説明］
- 照合は、BASICプログラムが正しく別のポケコンに記録されたか、あるいは別のポケコンから正しく読み込まれたか確認するために行います。
- 照合において、もし内容に不一致が生じたときはエラー82になります。
- BASICプログラムの照合が行われているときは画面右下に＊マークが表示されます。
照合が終われば＊マークは消え、プロンプト表示“>”になります。

ポケコン間通信命令

ＢＳＡＶＥ……バイナリ・セーブ
省略形……ＢＳ.
［プログラム］［マニュアル］

［機能］ ＢＡＳＩＣプログラムを別のポケコンに記録します。
［書式］ ＢＳＡＶＥ
［参照］ ＰＡＳＳ、ＢＬＯＡＤ、ＢＬＯＡＤ？
［説明］
- 別のポケコンに記録します。
- パスワードが設定されているときは、ＢＳＡＶＥ命令は無視されます。
- 実行するとオープンしていた全ファイルが閉じられます。

ポケコン間通信命令

ＢＳＡＶＥ　Ｍ…バイナリ・セーブ・エム
省略形………ＢＳ．Ｍ
［プログラム］［マニュアル］

［機能］ 機械語プログラムを別のポケコンに記録します。
［書式］ ＢＳＡＶＥ　Ｍ　開始番地，終了番地
［参照］ ＢＳＡＶＥ、ＢＬＯＡＤ　Ｍ
［説明］
- バイナリ形式で別のポケコンに記録します。
- 指定した開始番地から終了番地までの内容を書き込みます。
- 番地は10進数、または＆Ｈをつけて16進数で指定します。

基本命令

ＣＡＬＬ………コール
省略形……ＣＡ.
［プログラム］［マニュアル］

［機能］ 機械語プログラムを実行します。
［書式］ ＣＡＬＬ　番地
［参照］ ＰＥＥＫ、ＰＯＫＥ
［説明］
- 指定された番地から機械語プログラムを実行します。

　ＣＡＬＬ　＆Ｈ１Ｆ５８

（注）この命令を誤って使用するとＢＡＳＩＣプログラムや本機のシステムエリアを破壊し、異常が発生することがあります。この場合はリセットスイッチを押して、メモリの内容を消去してください。

関数

ＣＨＲ＄………キャラクタドル
省略形……ＣＨ.
［プログラム］［マニュアル］

［機能］ キャラクタコードを文字や記号など（キャラクタ）に変換します。
［書式］ ＣＨＲ＄　式
［参照］ ＡＳＣ
［説明］
- ＣＨＲ＄はＡＳＣ命令とは逆の関数で、キャラクタコードを文字や記号、数字に変換します。
  たとえば、キャラクタコード「９０」の文字を知りたい場合は
  　Ａ＄＝ＣＨＲ＄９０
  として実行すれば、Ａ＄には〝Ｚ〟が代入されます。
- 本機で扱うことのできるキャラクタと、それに対応するコードについては384ページのキャラクタ・コード表を参照してください。

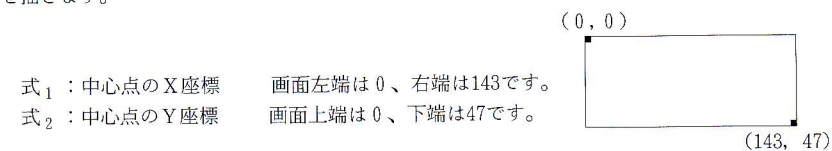
〈例〉
```
１０　ＡＡ＄＝〝〟
２０　ＩＮＰＵＴ〝コート゛＝〟；Ａ：ＣＬＳ
３０　ＡＡ＄＝ＡＡ＄＋ＣＨＲ＄Ａ
４０　ＬＯＣＡＴＥ　７，１：ＰＲＩＮＴ　ＡＡ＄
５０　ＧＯＴＯ　２０
```
このプログラムは、キャラクタコードを入力し、それを文字や記号に変換して文字変数ＡＡ＄に順次格納していきます。変換されたキャラクタはそのつど表示されます。
プログラムを実行させ、次のコードを入力してみてください。
　234、83、72、65、82、80（“◆ＳＨＡＲＰ”が表示されます）
256以上の値が指定されたときは、エラー33になります。

グラフィック命令

ＣＩＲＣＬＥ……サークル
省略形……ＣＩ.
［プログラム］［マニュアル］

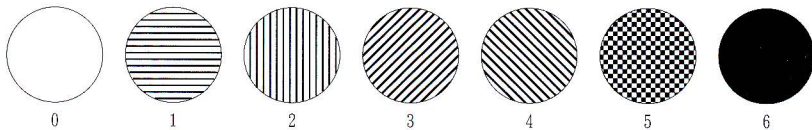
［機能］ ＤＥＧモードで円、扇形や楕円などを描きます。
［書式］ ＣＩＲＣＬＥ（式$_1$，式$_2$），式$_3$ ［，［式$_4$］，［式$_5$］，［式$_6$］［，［{Ｓ／Ｒ／Ｘ}］，［式$_7$］］］
［参照］ ＧＣＵＲＳＯＲ、ＬＩＮＥ、ＰＳＥＴ
［説明］
- （式$_1$，式$_2$）で指定した点を中心として、式$_3$で指定した長さ（ドット数）を半径とする円を描きます。

（0，0）　（143，47）

式$_1$：中心点のＸ座標　　画面左端は０、右端は143です。
式$_2$：中心点のＹ座標　　画面上端は０、下端は47です。

式$_1$、式$_2$の値は−32768〜32767の範囲内で指定できますが、上記の範囲を超えた場合は、画面外の指定の点を中心として円を描きます。
- 式$_3$で半径をドット数で指定します。式$_3$の範囲は１〜32767です。

● 式$_4$で開始角度、式$_5$で終了角度を度単位で指定します。
式$_4$、式$_5$は０～360度の範囲内で指定します。中心座標の右側が０で反時計方向に描きます。
省略した場合は、開始角度は０、終了角度は360が指定されます。（円になります。）
角度指定はＤＥＧモードにしてください。ＲＡＤ、ＧＲＡＤモードでは正しく描かれません。
● 開始角度、終了角度に負の値を指定したときは、円周から中心点までの線（半径線）を描きますので、扇形を描くことができます。負の値に対してはその絶対値が指定されたものとして描きます。
なお、０°の半径線を描くときで角度を変数で指定するときは、－360°と指定するか、もしくは－１＊変数名で指定してください。変数に－０を代入しても＋０として認識されます。
● 式$_6$の値により比率を指定します。
比率が１のときは円で、それ以外のときは楕円を描きます。省略した場合は１が指定されます。

$$比率=\frac{Y軸方向の半径（r_y）}{X軸方向の半径（r_x）}$$

比率を0.5にすると横長の楕円になります。
● Ｓ、Ｒ、Ｘで円周に相当するドットを点灯させるか、消すか、あるいは反転させるかを指定します。省略した場合はＳが指定されます。
Ｓ…円を描くとき、ドットを点灯させて円を描きます。（ドットをセット）
Ｒ…円を描くとき、ドットを消灯させて円を描きます。（ドットをリセット）
　円の回りのドットが点灯している場合に円を描くときや、描かれている円を消すときなどに用います。
Ｘ…円を描くとき、円に相当するドットが点灯しているときは消して、消えているときは点灯させます。（ドットの反転）
● 式$_7$で円の内部の模様を次のように指定します。省略した場合は０が指定されます。

〈例〉
```
10 CLS
20 CIRCLE(71,23),20,,,0.5,,2
30 END
```

横長の楕円を描きます

```
10 CLS
20 CIRCLE(71,23),20,-45,-135
30 END
```

扇形を描きます

（注）● 円の一部を描く場合でも、円のすべてが範囲内（－32768～32767）にないとエラー33になります。
● 画面はドットで構成されていますので、円、斜線、曲線などは正確な線にならない場合があります。
● 半径が小さい扇形を描くとき、描画精度が悪くなる場合があります。
● ＣＩＲＣＬＥ命令を実行するためには、ワーク用として約1440バイト以上のフリーエリアが必要です。

基本命令

ＣＬＥＡＲ……クリア　［プログラム］［マニュアル］
省略形……ＣＬ.

［機能］　固定変数の内容および配列変数、単純変数を消去します。
［書式］　ＣＬＥＡＲ
［参照］　ＤＩＭ
［説明］
● 単純変数や、ＤＩＭ命令により確保されていた配列変数はすべて消去（未定義の状態に）され、固定変数の内容もすべて消されます。
〈例〉 100 CLEAR:DIM B(4)
この例では、すべての変数を消去してから、配列変数Ｂ(４)の定義を行っています。プログラムの先頭ではよくこのような方法を用います。

ファイル関連命令

ＣＬＯＳＥ……クローズ　［プログラム］［マニュアル］
省略形……ＣＬＯＳ.

［機能］　入出力のファイルを閉じます。
［書式］　ＣＬＯＳＥ［＃ファイル番号，＃ファイル番号……］
［参照］　ＯＰＥＮ、ＰＲＩＮＴ＃、ＩＮＰＵＴ＃
［説明］
● ファイル番号に対応するファイルを閉じます。ファイル番号は１、２または３のいずれかです。
● ファイル番号を省略するとすべてのファイルを閉じます。
● ファイルが出力（ＯＵＴＰＵＴまたはＡＰＰＥＮＤ）モードでオープンしていた場合は、メモリ（出力バッファ）に残っているデータおよびファイル終了コードを、出力してからクローズします（閉じます）。
● 次の場合も自動的にファイルを閉じます。
・ＥＮＤ、ＮＥＷ、ＲＵＮ命令を実行したとき
・電源が切れたとき
・プログラムの編集をしたとき（プログラムの入力、修正、削除やＤＥＬＥＴＥ命令などを実行したとき）
・プログラムファイルエリアへのプログラムの登録、呼び出し、削除などを行ったとき
・ＢＳＡＶＥ、ＢＬＯＡＤ、ＢＬＯＡＤ？、ＢＳＡＶＥ　Ｍ、ＢＬＯＡＤ　Ｍ命令実行時
・機械語モニタモードへ切り替えたとき
・ＢＡＳＩＣモード（ＲＵＮ、ＰＲＯ）から他のモード（ＴＥＸＴ、ＣＡＳＬなど）へ切り替えたとき

基本命令

ＣＬＳ…………クリアスクリーン　［プログラム］

省略形……なし

［機能］ 表示内容を消去します。
［書式］ ＣＬＳ
［参照］ ＬＯＣＡＴＥ
［説明］ 表示内容を消去し、表示開始位置を（0，0）位置に戻します。

基本命令

ＣＯＮＴ………コンティニュー　［マニュアル］

省略形……Ｃ.

［機能］ 一時停止しているプログラムの実行を再開します。（ＲＵＮモードのマニュアル操作でのみ有効）
［書式］ ＣＯＮＴ［↵］
［参照］ ＳＴＯＰ
［説明］ ＳＴＯＰ命令や［BREAK ON］によりプログラムが一時停止しているとき、実行を再開させます。

基本命令

ＤＡＴＡ………データ　［プログラム］

省略形………ＤＡ.

［機能］ ＲＥＡＤ文に続く変数に与えるデータを指定します。
［書式］ ＤＡＴＡ ｛式／文字列｝，｛式／文字列｝……
［参照］ ＲＥＡＤ
［説明］
- ＲＥＡＤ命令の説明を参照ください。

基本命令

ＤＥＧＲＥＥ…ディグリー　［プログラム］［マニュアル］

省略形……ＤＥ.

［機能］ 角度単位を“度”に設定します。
［書式］ ＤＥＧＲＥＥ
［参照］ ＲＡＤＩＡＮ、ＧＲＡＤ
［説明］
- 三角関数、逆三角関数、座標変換で扱う角度の単位を“度”単位〔°〕に設定します。（1直角＝90°）

基本命令

ＤＥＬＥＴＥ…デリート　［マニュアル］

省略形……ＤＥＬ.

［機能］ プログラム行を削除します。（ＰＲＯモードのマニュアル操作でのみ有効）
［書式］ （1）ＤＥＬＥＴＥ　開始行番号［－［終了行番号］］［↵］
（2）ＤＥＬＥＴＥ　－終了行番号［↵］
［参照］ ＮＥＷ、ＰＡＳＳ
［説明］
- 開始行番号と終了行番号を指定したときは、その間に含まれるすべての行を削除します。
- 開始行番号のみを指定したときは、その行だけを削除します。
- 開始行番号とハイフン（－）を指定したときは、開始行番号以降のすべての行を削除します。
- 書式（2）では、プログラムの先頭行から終了行番号間に含まれるすべての行を削除します。
- 開始行番号と終了行番号の両方を省略したときは、エラー10になります。
- 指定した行番号が存在しないときは、エラー40になります。
- 開始行番号が終了行番号よりも大きい指定をすると、エラー44になります。
- パスワードが設定されているときは、ＤＥＬＥＴＥ命令は無視されます。

基本命令

ＤＩＭ…………ディメンジョン　［プログラム］［マニュアル］

省略形……Ｄ.

［機能］ 配列名と、その大きさを定義（宣言）し、配列変数をメモリ（プログラム・データエリア）上に確保します。
［書式］ （1）ＤＩＭ ｛配列名（$式_1$）／配列名（$式_1$，$式_2$）｝［，｛配列名（$式_1$）／配列名（$式_1$，$式_2$）｝…］
（2）ＤＩＭ ｛配列名（$式_1$）／配列名（$式_1$，$式_2$）｝＊$式_3$［，｛配列名（$式_1$）／配列名（$式_1$，$式_2$）｝＊$式_3$…］
（注）書式（2）は文字変数でのみ使用できます。
［参照］ ＣＬＥＡＲ、ＲＵＮ、ＥＲＡＳＥ
［説明］
- 配列変数を使用するときは、事前にＤＩＭ命令により配列名と、その大きさを定義（宣言）して、メモリ（プログラム・データエリア）上に確保しておく必要があります。
- 配列名はアルファベット１文字あるいは２文字（２文字目は数字も使用可能）で指定し、文字配列変数の場合は後に＄マークをつけます。
- $式_1$ および$式_2$ は添字といわれ、配列の大きさ（配列要素数）と次元を指定します。添字が１つのものを一次元配列と呼び、２つのものを二次元配列と呼びます。本機では二次元配列までが使用できます。

〈例〉　ＤＩＭ　Ｂ（3）　　一次元配列変数Ｂ（　）について、配列要素Ｂ(0)、Ｂ(1)、Ｂ(2)、Ｂ(3）の４個が確保されます。

ＤＩＭ　ＸＡ＄（2，3）　　二次元配列変数ＸＡ＄（　）について、配列要素ＸＡ＄(0，0)、ＸＡ＄(0，1)、………ＸＡ＄(2，2)、ＸＡ＄(2，3）の12個が確保されます。

添字は理論的に0～255までの整数値を用いることができますが、計算機のメモリの大きさ、使用状態によっては添字で指定しただけ、変数が確保できない場合があります。（確保できないときはエラー60になります。）
- 添字が小数部を含んでいるときは小数部は無視され、整数部のみが有効になります。
- 添字は数値変数や式の形で用いることもできます。

〈例〉
```
10 INPUT A, B
20 DIM X(A), Y(B-1)
```
- 文字配列変数は変数の長さを指定できます。

書式（２）において、式$_3$により文字数を１～255文字の範囲で任意に指定できます。

〈例〉　ＤＩＭ　Ｆ＄（２）＊３０　　Ｆ＄（０）～Ｆ＄（２）の各変数には、それぞれ最大30文字まで記憶できます。

　　　ＤＩＭ　Ｙ＄（５，４）＊６　　Ｙ＄（０，０）～Ｙ＄（５，４）の各変数には、それぞれ最大６文字まで記憶できます。

文字数の指定（＊式$_3$）を省略した場合は、自動的に16文字が指定されます。
指定文字数が大きいほど多くのメモリを必要とします。

● 複数の配列を使用する場合は、ＤＩＭ命令で一度に定義することができます。
〈例〉　ＤＩＭ　Ｊ（５），Ｋ＄（４，３），ＸＢ＄（５）＊１０

● 一度定義した配列名は再定義できません。
たとえば、ＤＩＭ　Ｘ（５）とＤＩＭ　Ｘ（３，４）は同じＸという配列名になり、同時に使用できません。ただし、数値配列変数と文字配列変数は別の配列とみなされます。たとえば、配列Ｚ（　）とＺ＄（　）は同時に使用できます。

● 配列変数はＥＲＡＳＥ、ＣＬＥＡＲ命令により消去する（未定義の状態にする）ことができます。
また、ＲＵＮ命令によりプログラムの実行を開始したときも、以前に定義されていた配列変数はすべて消去されます。ＧＯＴＯ命令によるプログラムの実行では、変数は消去されません。
したがって、一度実行したプログラムをＧＯＴＯ命令により再実行させるときなどに、ＤＩＭ命令の書かれている行を実行させると同じ変数名を再定義することになり、エラー30になります。このような場合はＥＲＡＳＥまたはＣＬＥＡＲ命令で消去してから、定義し直すようにしてください。
〈例〉
```
50 "S":ERASE X: DIM X(3,4)
```

基本命令

ＤＭＳ＄………ディー・エム・エス・ドル　［プログラム］［マニュアル］
省略形……ＤＭ.

［機能］　10進数（度）を60進数（度・分・秒）の文字列に変換します。
［書式］　ＤＭＳ＄　式
［参照］　ＶＤＥＧ
［説明］
● 10進数を60進数の文字列（度（°）・分（′）・秒（″）の記号を含む）の形に変換します。
〈例〉
```
10 B=61.01
20 AA$=DMS$ B
30 PRINT AA$
```
```
RUN
61° 00′ 36″
>
```

（注）変換した結果を格納する文字変数には、文字単純変数か文字配列変数を使用してください。文字固定変数には、最大７文字まで格納できます。（変数の長さについては153ページを参照）

〈例〉
```
10 B=1.2345
20 A$=DMS$ B
30 PRINT"A$= ";A$
40 AA$=DMS$ B
50 PRINT"AA$=";AA$
```
```
RUN
A$= 1° 14′ 04
AA$=1° 14′ 04.2″
>
```

● 度・分・秒の記号のキャラクターコード（くわしくは384ページ参照）
度（°）…223（Ｈ＆ＤＦ）、分（′）…39（Ｈ＆２７）、秒（″）…248（Ｈ＆Ｆ８）

基本命令

ＥＮＤ…………エンド　［プログラム］
省略形……Ｅ.

［機能］　プログラムの実行を終了させます。
［書式］　ＥＮＤ
［説明］　● プログラムの実行を終了し、オープンされていたファイルをすべて閉じます。

基本命令

ＥＯＦ……エンド・オブ・ファイル　［プログラム］［マニュアル］
省略形……ＥＯ.

［機能］　ファイルの終わりを検出します。
［書式］　ＥＯＦ（ファイル番号）
［参照］　ＯＰＥＮ
［説明］
● ファイル番号で指定したファイルのデータを最後まで読み取ったかどうかを調べます。ファイル番号は１、２、３のいずれかです。
● ラムデータファイルまたは11ピンの通信時のみ有効です。
● 最後に達していれば－１（真）、そうでなければ０（偽）の値を与えます。
● 11ピン通信時で全二重通信の場合は、受信バッファ内が空のときは－１（真）、空でないときは０（偽）になります。
半二重通信の場合は、最後に達していれば－１（真）、そうでなければ０（偽）になります。
● ファイル番号で指定したファイルの装置名がＥ（ラムデータファイル）の場合は、ＩＮＰＵＴモードでオープンされていないとエラーになります。
〈例〉　ラムデータファイル（ＡＢＣ．ＤＡＴ）の確保（181ページ参照）をした後で、次のプログラムを入力してください。
```
10 OPEN"E:ABC.DAT" FOR OUTPUT AS #2
20 PRINT #2,123,456,789
30 CLOSE
40 OPEN"E:ABC.DAT" FOR INPUT AS #2
50 INPUT #2,A,B
60 X=EOF(2)          ← データの読み出しは終了していない
70 INPUT #2,C
80 Y=EOF(2)          ← データの読み出しは終了している
90 CLOSE :PRINT X,Y:END
```
このプログラムを実行すると、Ｘ＝０、Ｙ＝－１になります。

基本命令

ＥＲＡＳＥ……イレーズ　［プログラム］［マニュアル］
省略形……ＥＲ.

［機能］　配列変数を消去します。
［書式］　ＥＲＡＳＥ 配列名［，配列名……］

参照　ＣＬＥＡＲ、ＤＩＭ

説明
- 指定した配列変数を消去します。
- ＥＲＡＳＥ命令で配列変数を消去すれば、その配列変数が占めていた領域を他の目的のために使うことができます。
  また、配列の大きさを変更したいとき、ＥＲＡＳＥ命令でそれまでの配列を消去すればＤＩＭ命令で配列の大きさを再定義できます。
- すべての変数を消去するときはＣＬＥＡＲ命令を使います。

〈例〉
```
 10 DIM A(2,3),B$(5)*30
  ⋮
100 ERASE B$
```

ファイル関連命令

ＦＩＬＥＳ……ファイルズ
省略形……ＦＩ.　［マニュアル］

機能　プログラムファイルエリアに登録されているファイルのファイル名とファイルサイズを表示します。

書式　ＦＩＬＥＳ［↵］

参照　ＬＦＩＬＥＳ、ＳＡＶＥ、ＬＯＡＤ

説明
- 実行すると先頭のファイル名から表示します。画面には６つのファイル名まで表示されます。
- ファイル名の左側に➡マークが点灯しています。➡マークは［▲］、［▼］で上下に移動できます。これを移動させていけば必要に応じて画面が送られて、別のファイル名を表示します。
  ➡マークを必要なプログラムのファイル名に移動させてから［SHIFT］＋［M］(LOAD) を押すと、そのプログラムを呼び出すことができます。ただし、ＴＥＸＴモードで登録したプログラムを呼び出すことはできません。
- ファイルサイズについてのくわしい説明は、179ページをご覧ください。
- ［CLS］、［BREAK］などでファイル名の表示を解除できます。

基本命令

ＦＩＸ……フィックス
省略形……なし　［プログラム］［マニュアル］

機能　数値の整数部を求めます。

書式　ＦＩＸ　式

説明
- 式の値の小数点以下を取り除いた整数部だけの値を求めます。

〈例〉
```
A=FIX 2.5         Aには2が代入されます。
A=FIX -2.5        Aには-2が代入されます。
A=FIX(2.45*3)     Aには7が代入されます。
```

基本命令

ＦＯＲ～ＮＥＸＴ…フォー～ネクスト
省略形……Ｆ.　Ｎ.　ＳＴＥ.　［プログラム］

機能　ＦＯＲとＮＥＸＴの間に書かれた命令を指定された条件が満たされるまで、繰り返し実行します。

書式　ＦＯＲ　数値変数＝初期値　ＴＯ　最終値　ＳＴＥＰ　きざみ値
　　　　～
　　　ＮＥＸＴ［数値変数］

参照　ＲＥＰＥＡＴ～ＵＮＴＩＬ、ＷＨＩＬＥ～ＷＥＮＤ

説明
- 数値変数が初期値から始まって、指定されたきざみ値分ずつ増加（あるいは減少）していき、最終値よりも大きく（あるいは小さく）なるまでＦＯＲとＮＥＸＴの間を繰り返し実行します。（この繰り返し部分をＦＯＲ～ＮＥＸＴループと呼びます。）

〈例〉
```
FOR A=0 TO 10 STEP 2
  ~
NEXT A
```
Ａが０から始まって、１回ＦＯＲとＮＥＸＴ間を実行するごとにＡに２を加えながら、Ａの値が10を超えるまで、ＦＯＲとＮＥＸＴの間を実行します。

- きざみ値が１のときはＳＴＥＰ　１を省略することができます。
- ＦＯＲとＮＥＸＴは必ず対にして使い、ＦＯＲの後の数値変数とＮＥＸＴの後の数値変数は同一でなければなりません。ただし、ＮＥＸＴの後の数値変数は省略できます。

```
FOR B=1 TO 5
  ~               ← 同じ数値変数にする。
NEXT B
```

- 初期値、最終値、きざみ値（ステップ値）は次の範囲内で指定できます。
  −9.999999999Ｅ99～9.999999999Ｅ99
  $(-9.999999999\times 10^{99} \sim 9.999999999\times 10^{99})$
- 初期値にきざみ値を加えると最終値から離れてしまう場合は、ループ内を１回だけ実行してループを抜けます。
  なお、きざみ値を０に指定しますと、永遠にループ内の実行を繰り返すプログラムになってしまいます。
- ＦＯＲ～ＮＥＸＴループの中に、別のＦＯＲ～ＮＥＸＴループを入れる場合、中に入るＦＯＲ～ＮＥＸＴループは外のＦＯＲ～ＮＥＸＴループ内に完全に入っていなければなりません。この条件でループを最大５段まで重ねて使う（深みをもたせる）ことができます。（379ページのスタック参照）
- ＦＯＲ～ＮＥＸＴループの外からループ内に飛び込むことはできません。（飛び込ませるとエラー52になります。）

（注）・ＦＯＲ～ＮＥＸＴループから外に飛び出した場合、そのループは終了したことになりません。プログラムによっては（ＦＯＲ命令を何回か実行するようなプログラムの場合）、ＦＯＲ～ＮＥＸＴの深みエラー50が発生することがあります。
　　　・ＦＯＲ～ＮＥＸＴループ内ではＣＬＥＡＲ、ＥＲＡＳＥ、ＤＩＭ命令は使用できません。

関数

ＦＲＥ…………フリー
省略形……ＦＲ.　［プログラム］［マニュアル］

機能　未使用部分のバイト数を求めます。

書式　ＦＲＥ

説明
- 本機内のメモリの中で、現在使用されていない部分のバイト数が得られます。
  ＢＡＳＩＣのプログラムや配列変数、単純変数および機械語エリア、プログラムファイルエリア、ラムデータファイルエリア、テキストエリアとして使用されている部分以外のバイト数を求めます。

グラフィック命令

GCURSOR………グラフィックカーソル
省略形……GC.

プログラム　マニュアル

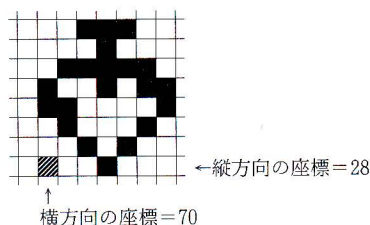
機能　グラフィック表示の開始位置をドット（点）単位で指定します。

書式　GCURSOR（式$_1$，式$_2$）

参照　GPRINT、LOCATE

説明
- 画面は横144、縦48のドット（点）で構成されています。
このドットはそれぞれ横方向に0～143、縦方向に0～47の番号がつけられています。この番号をX－Y座標と同様の形、つまり式$_1$でX方向、式$_2$でY方向の番号を指定して、表示開始位置（表示の開始ドット）を指定します。

- 式$_1$、式$_2$は－32768～32767の範囲で指定できます。ただし、画面をはみ出すような指定を行う（式$_1$が0～143、式$_2$が0～47の範囲外の値）と、画面外の仮想位置を表示開始位置に指定することになります。
- 表示開始位置はRUN命令を実行したときや電源を入れ直したとき、または、SHIFT＋CAを押したときなどでは（0，7）位置が指定されます。

〈例〉
```
10 CLS :WAIT
20 GCURSOR(70,28)
30 GPRINT"1834458F452C18"
```
このプログラムを実行すると、画面の中程に次のように表示されます。
（斜線部分は表示されません。）

基本命令

GOSUB～RETURN…ゴーサブ～リターン
省略形……GOS.　RE.

プログラム

機能　指定した行から始まるサブルーチンへプログラムの実行を移し、RETURN命令で戻ります。

書式　GOSUB {行番号 / ラベル}
RETURN

参照　GOTO、ON　GOSUB

説明
- 何回も同じ計算や処理が出てくる場合、その部分を抜き出してプログラムしておきます。抜き出したプログラムを必要に応じて実行すれば、プログラムを短かく簡略化できます。
- サブルーチンへのジャンプはGOSUB命令に続いて、サブルーチンのおかれている行番号またはラベルを書いて指示します。（ラベルについては163ページ参照）

〈例〉
```
        ⋮
  50:GOSUB 200      200行へサブルーチンジャンプ
        ⋮
 100:GOSUB"A"       ラベル"A"または*Aが書かれている行へ
        ⋮           サブルーチンジャンプ
```

- サブルーチンの最後にはRETURN命令を書いて、メインルーチンへの復帰を指示します。（RETURN命令には行番号の指定は不要です。）
メインルーチンへ復帰したときは、GOSUB命令の次の命令を引き続き実行します。
- サブルーチンから別のサブルーチンへ実行を移すことができます。
サブルーチンからサブルーチンへ、また次のサブルーチンへ……というように重ねて使用する場合、最高10段まで重ねる（深みをもたせる）ことができます。（10段を超えるとエラー50になります。）

基本命令

GOTO………ゴートゥー
省略形……G.

プログラム　マニュアル

機能　プログラムの実行を指定された行へ無条件に移します。

書式　GOTO {行番号 / ラベル}

参照　GOSUB、ON～　GOTO、RUN

説明
- 通常プログラムは小さい行から順次実行されますが、GOTO命令により、その実行を指定した行へ移す（ジャンプさせる）ことができます。
また、RUNモードでのマニュアル操作により、指定した行からプログラムの実行を開始させることができます。
- ジャンプ先は、GOTO命令に続けて行番号またはラベルを書いて指定します。（ラベルについては163ページ参照）

〈例〉
```
GOTO 40      40行へジャンプしなさい。
GOTO "AB"    ラベル"AB"または*ABのついている行へジャンプしなさい。
```

- 指定した行番号およびラベルがない場合はエラー40になります。
- 同じラベルが2個以上書かれているときは、行番号の小さいほうへジャンプします。
- GOTO命令によりプログラムの実行が開始されたときの計算機の状態については163ページを参照してください。

グラフィック命令

GPRINT………グラフィックプリント
省略形……GP.

プログラム　マニュアル

機能　指定されたドットパターンを表示します。

書式　（1）ＧＰＲＩＮＴ　文字列

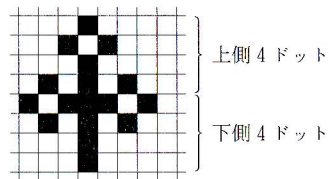
　　　（2）ＧＰＲＩＮＴ　式［；式；式；....］

参照　ＧＣＵＲＳＯＲ、ＰＲＩＮＴ

説明
- 文字列や式で、表示させるドットパターンを指定します。ドットパターンは縦に並ぶ８ドットを１まとまりとして指定します。
- 書式（１）では、縦に並ぶ８ドットを下側４ドットと上側４ドットに分けて、それぞれのパターンを16進数で表し、文字列として″　″で囲んで指定します。

〈例〉　ＧＰＲＩＮＴ″○○ ○○ ○○.... ″

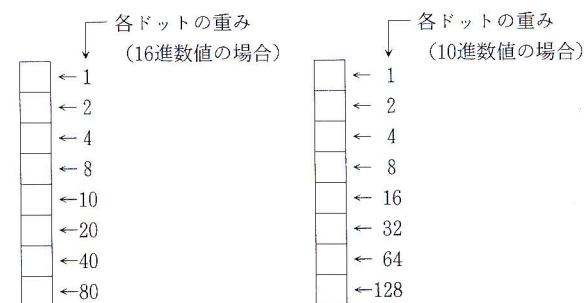
数字2字を1組にして縦1列のドットパターンを指定します。最初の数字は下側4ドット、2番目の数字は上側4ドットを表します。

〈例１〉　ＧＰＲＩＮＴ″１０２８１２ＦＤ１２２８１０″

０８２Ｄ２８０……上側のドットパターンを16進数で表した場合

１２１Ｆ１２１……下側のドットパターンを16進数で表した場合

- 書式（２）では、縦に並ぶ８ドットを１まとまりとして、ドットパターンを数値で指定します。縦に並ぶ８ドットは、下に示すようにそれぞれ“重み”が与えられています。

各ドットの重み（16進数値の場合）：←1　←2　←4　←8　←10　←20　←40　←80

各ドットの重み（10進数値の場合）：←1　←2　←4　←8　←16　←32　←64　←128

ドットパターンを指定するときは、点灯させたいドットの重みを加え合わせた値で指定します。

〈例２〉　例１に示したパターンを表示します。

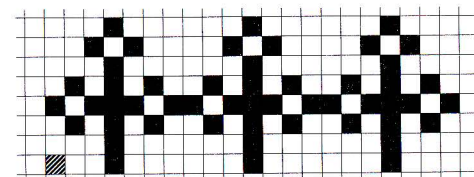
16進数値による指定

ＧＰＲＩＮＴ　＆Ｈ１０；＆Ｈ２８；＆Ｈ１２；＆ＨＦＤ；＆Ｈ１２；＆Ｈ２８；＆Ｈ１０

10進数値による指定

ＧＰＲＩＮＴ　１６；４０；１８；２５３；１８；４０；１６

- ＧＣＵＲＳＯＲ命令で表示開始位置が指定されている場合、ＧＰＲＩＮＴ命令で表示される内容は、指定されている表示開始位置（ドット）を含んだ上側の８ドットを使って、最初のドットパターンを表示します。

〈例３〉
```
１０　ＡＡ＄＝″１０２８１２ＦＤ１２２８１０″
２０　ＧＣＵＲＳＯＲ（３０，２０）
３０　ＧＰＲＩＮＴ　ＡＡ＄；ＡＡ＄；ＡＡ＄
```

ＧＣＵＲＳＯＲ命令の指定位置（３０，２０）
この位置より上側８ドットを使用して表示が開始されます。

- ＧＰＲＩＮＴ命令がセミコロン（；）で終わっている場合は、実行後、表示された内容の１列右側が次の表示開始位置に指定されます。なお、ＧＰＲＩＮＴ命令の最後がコロン、または改行の場合は、横方向（Ｘ方向）の指定が０に戻ります。
- ＷＡＩＴ命令で指定された時間だけ表示します。

基本命令

ＧＲＡＤ………グラード　　プログラム　マニュアル

省略形……ＧＲ.

機能　角度単位を“ＧＲＡＤ”（グラード）に設定します。

書式　ＧＲＡＤ

参照　ＤＥＧＲＥＥ、ＲＡＤＩＡＮ

説明　● 三角関数、逆三角関数、座標変換で扱う角度の単位を“ＧＲＡＤ”単位［g］に設定します。（１直角＝100^{g}）

基本命令

ＨＥＸ＄……ヘキサドル

省略形……Ｈ.

プログラム　マニュアル

機能　数値データを10進数とみなし、16進数の文字列に変換します。

書式　ＨＥＸ＄　式

説明　式の値は－9999999999から9999999999の範囲で、整数部のみが有効です。

〈例〉　Ｃ＄＝ＨＥＸ＄12＋ＨＥＸ＄15　Ｃ＄には"ＣＦ"が代入されます。

基本命令

ＩＦ～ＴＨＥＮ～ＥＬＳＥ…イフ～ゼン～エルス

省略形……ＩＦ　Ｔ.　ＥＬ.

プログラム

機能　条件を判断し、プログラムの流れ（実行の順番）を変えます。

書式　ＩＦ　条件式　ＴＨＥＮ {行番号 / ラベル / 実行文} ［ＥＬＳＥ {行番号 / ラベル / 実行文}］

参照　ＡＮＤ、ＯＲ

説明
- ＩＦに続く条件式が成立した場合はＴＨＥＮに続く命令を実行します。条件式が成立しない場合は、ＥＬＳＥに続く命令を実行します。ＥＬＳＥを省略した場合、条件式が成立しないときは次の行へ実行が移ります。
- ＴＨＥＮやＥＬＳＥに続けて行番号またはラベルを書くと、その行またはラベルが書かれている行へジャンプします。
- 実行文として代入文を書くときは、ＴＨＥＮまたはＬＥＴ命令が必要です。
- ＴＨＥＮ命令を省略して実行文（ＰＲＩＮＴ文、ＩＮＰＵＴ文など）を書くことができますが、ジャンプを指示する場合はＴＨＥＮまたはＧＯＴＯ命令が必要です。（ＥＬＳＥは省略できません。）

  具体的な使いかたについては、117ページも参照してください。

〈例〉　ＩＦ　Ａ＜５　ＴＨＥＮ　Ｃ＝Ａ＊Ｂ：ＧＯＴＯ　５０

もしＡが５よりも小さければ、Ａ＊Ｂの結果をＣに代入して、50行へジャンプします。

ＩＦ　Ｂ＝Ｃ＋１　ＧＯＴＯ　６０　ＥＬＳＥ　１００
または　ＩＦ　Ｂ＝Ｃ＋１　ＴＨＥＮ　６０　ＥＬＳＥ　１００

もし、ＢとＣ＋１の結果が等しければ60行へジャンプし、等しくなければ100行へジャンプします。

- 条件式としては、＝、＞、＞＝、＜、＜＝、＜＞が使用できます。
- 条件式は、ＡやＢ＋４のような変数や、通常の式にすることもできます。

  ＩＦ　Ｂ＋４　ＴＨＥＮ　６０
  ＩＦ　Ａ　ＴＨＥＮ　６０

  この場合は、変数や式の値が０以外（ＩＦ文が成立）ならばＩＦ文の後に続く内容を実行し、０（ＩＦ文が不成立）ならば次の行を実行します。

- 条件式は次の例のように“＊”と“＋”を用いて結合できます。

〈例〉　ＩＦ（Ａ＞５）＊（Ｂ＞１）ＴＨＥＮ…………

Ａが５よりも大きく、かつＢが１よりも大きいとき、ＴＨＥＮに続く内容を実行（論理積：ＡＮＤ）

ＩＦ（Ａ＞５）＋（Ｂ＞１）ＴＨＥＮ…………

Ａが５よりも大きいか、あるいはＢが１よりも大きいとき、ＴＨＥＮに続く内容を実行（論理和：ＯＲ）

- 文字列の比較

  条件式に文字列を用いることにより、文字列の比較、大小判断ができます。
  文字列はキャラクタコードの大きさによって比較されます。たとえば、キャラクタコードでは“Ａ”が65、“Ｂ”が66、“Ｃ”が67……となっています。したがって“Ａ”は“Ｂ”よりも小さく、“Ｂ”は“Ｃ”よりも小さくなります。同様に数字はアルファベットより小さくなります。

〈例〉　150行のデータを読み込んで、アイウエオ順に並べ替えます。

```
  5 CLS
 10 DIM Y$(4)
 20 READ Y$(1),Y$(2),Y$(3),Y$(4)
 30 GOSUB 200:PRINT
 40 FOR A=1 TO 3                          ┐
 50 FOR B=A+1 TO 4                        │
 60 IF Y$(A)<=Y$(B) THEN 80               │ 比較、並べ替えループ
 70 Y$(0)=Y$(A):Y$(A)=Y$(B):              │
    Y$(B)=Y$(0)                           │
 80 NEXT B:NEXT A                         ┘
 90 GOSUB 200:END
150 DATA アサミ,サチコ,アサコ,キヨミ
200 FOR A=1 TO 4
210 PRINT Y$(A);"␣␣";
220 NEXT A
230 PRINT:RETURN
```

- １つの式で扱う文字列の長さについて

  文字列の結合や大小比較などの式において、１つの式に含まれる文字（アルファベット、カタカナ、数字など）の数は、合計で255文字を超えない範囲で演算処理を行ってください。255文字を超えると、エラー55になります。

基本命令

ＩＦ～ＴＨＥＮ～ＥＬＳＥ～ＥＮＤＩＦ…イフ～ゼン～エルス～エンドイフ

省略形……ＩＦ　Ｔ.　ＥＬ.　ＥＮＤＩ.

プログラム

機能　条件を判断し、プログラムの流れ（実行の順番）を変えます。

書式　ＩＦ　条件式　ＴＨＥＮ
　　　実行文１
　　［ＥＬＳＥ

```
    実行文２]
ＥＮＤＩＦ
    実行文３
```

参照　ＩＦ～ＴＨＥＮ～ＥＬＳＥ、ＡＮＤ、ＯＲ、ＮＯＴ、ＸＯＲ

説明
- ＩＦ、ＥＬＳＥ、ＥＮＤＩＦはそれぞれ行頭（ラインナンバーのすぐ後ろ）に書く必要があります。なお、ラベルの後ろは行頭にはなりません。
- ＴＨＥＮ（もしくはＥＬＳＥ）の後ろにステートメント、式、リマーク等の実行文を書くことはできません。ＴＨＥＮの後ろにステートメントなどを書いたり、ＴＨＥＮを省略したりすると１行形式のＩＦ～ＴＨＥＮ～ＥＬＳＥと見なされます。
- 条件式が成立した場合は、実行文１を実行した後、実行文３に移ります。
  条件式が成立しない場合は、実行文２を実行した後、実行文３に移ります。
- ＥＬＳＥと実行文２を省略した場合、条件式が成立しないときは、実行文３に移ります。
- 条件式の使用方法や判断内容はＩＦ～ＴＨＥＮ～ＥＬＳＥ（１行形式）の場合と同じです。

〈例〉　鶴亀算を行うプログラムです。

```
 10 WAIT:CLS
 20 INPUT"ｾﾞﾝﾀｲﾉｶｽﾞ:";A
 30 LOCATE 14,0:INPUT"ｱｼﾉｶｽﾞ:";B
 40 IF (4*A)<B OR (2*A)>B THEN    ─┐
 50  PRINT"IMPOSSIBLE ﾔﾘﾅｵｼ!"        │
 60 ELSE                            │
 70  C=B-INT(B/2)*2                 │
 80   IF C=1 THEN       ─┐          │
 90    PRINT"ｱｼﾉｶｽﾞｶﾞ ｷｽｳﾃﾞｽ"  │          │
100   ELSE               │②        │①
110    X=(2*A)-B/2:Y=(B/2)-A    │          │
120    WAIT 0:PRINT"ﾂﾙﾉｶｽﾞﾊ";X  │          │
130    WAIT:PRINT"ｶﾒﾉｶｽﾞﾊ ";Y   │          │
140   ENDIF             ─┘          │
150 ENDIF                         ─┘
160 GOTO 10
```

①と②でブロック形式のＩＦ～ＴＨＥＮ～ＥＬＳＥ～ＥＮＤＩＦを使用しています。

関数

ＩＮＫＥＹ＄…インキードル　〔プログラム〕

省略形……ＩＮＫ.

機能　押されたキーの内容を読み込んで指定された文字変数に代入します。

書式　文字変数＝ＩＮＫＥＹ＄

説明
- 通常、次の例のように繰り返しループを作って、有効なキーが押されるのを待ちます。実行されたときに、キーが押されていなければ、変数にはNull（空白）が代入されます。

〈例〉

```
10 CLS
20 Z$=INKEY$            ] このラインを繰り返し実行し、
30 IF Z$=""THEN 20      ] キーが押されるのを待ちます。
40 PRINT"---";ASC Z$;"---"
50 Z$=INKEY$
60 IF Z$=""THEN 10 ELSE 50
```


- 〔BREAK ON〕はプログラムの一時停止キー（ブレークキー）として働きます。
- 〔SHIFT〕を押しながら、キーを押したときに働く機能や入力される記号、〔2nd F〕に続いて押したときに働く機能などを読み込むことはできません。また、英小文字やカナ文字を読み込むこともできません。
- 〔OFF〕、〔SHIFT〕は読み込むことはできません。
- 命令実行時に押されていたキーのデータを読み込むための命令です。ＩＮＰＵＴ命令のようにキー入力時に〔↵〕を押す必要はありません。

（注）プログラムの初めにＩＮＫＥＹ＄があると、プログラムをスタートさせたときにスタートキーを読み取ってしまうことがあります。

ＩＮＫＥＹ＄で読み取られるキーとキーコード

		0	16	32	48	64	80	…	128	144	…	240
	16進上位／16進下位	0	1	2	3	4	5	…	8	9	…	F
0	0		2nd F	SPACE	0		P					
1	1				1	A	Q			ln		
2	2	CLS			2	B	R			log		
3	3				3	C	S					
4	4	▲	カナ		4	D	T					
5	5	▼	CAPS		5	E	U			sin		
6	6				6	F	V			cos		
7	7	ANS	BS		7	G	W		$1/x$	tan		
8	8	BASIC	R•CM	(	8	H	X		x^2			
9	9	TEXT	M+	)	9	I	Y					
10	A	TAB		*		J	Z					
11	B	INS		+	;	K				→DEG		π
12	C	CONST		,		L				F↔E		$\sqrt{\ }$
13	D	↵		−	=	M				${}_nP_r$		
14	E	►		.		N	y^x∧			MDF		
15	F	◄		/		O						

基本命令

ＩＮＰＵＴ……インプット　〔プログラム〕

省略形……Ｉ.

機能　キーから数値または文字列の入力を行います。

書式
（１）ＩＮＰＵＴ　変数［，変数……］
（２）ＩＮＰＵＴ"文字"，変数［，"文字"，変数……］
（３）ＩＮＰＵＴ"文字"；変数［，"文字"；変数……］

参照　ＩＮＰＵＴ＃、ＩＮＫＥＹ＄、ＲＥＡＤ、ＬＯＣＡＴＥ

説明
- 変数に、キーボードから数値や文字列を代入したいときに使用します。ＩＮＰＵＴ命令に続いて、データを格納するための変数を指定します。
- 書式（１）では、？を表示して入力待ち（プログラムの実行を停止）になります。データを入

力して、⏎を押せば、データが変数に代入されて実行が再開されます。
（変数を複数個指定する場合は コンマ（，）で区切ります。）

● 書式（２）では、"　"で囲まれた文字を入力ガイダンス（入力案内）として表示し、入力待ちになります。
データを入力すると入力ガイダンスは消えます。
［入力ガイダンスは、計算機がデータ待ちになったとき、何のデータを要求しているのか、どのデータを入力すればよいか、などをわかりやすくするためのメッセージです。］
● 書式（３）では、書式（２）の場合と同じく入力ガイダンスを表示して入力待ちになります。
データを入力すると、入力ガイダンスに続けて、入力したデータが表示されます。
● 書式（1）、（2）、（3）は１つのＩＮＰＵＴ命令の中で同時に使えます。
〈例〉　ＩＮＰＵＴ　"Ａ＝"；Ａ，Ｂ，"Ｃ＝？"，Ｃ
● ＩＮＰＵＴ命令の入力待ちのときに、データを入力せずに⏎のみを押したときは、変数に入っていたデータを保持したまま、次の実行に移ります。
● ＩＮＰＵＴ命令に続いて指定された変数の型と、入力するデータの型は同じでなければなりません。
文字変数に数値を入れると文字データとして認識されます。数値変数に文字を入れると、その変数に記憶されている数値が入力されます。
● ＩＮＰＵＴ命令実行前に、ＬＯＣＡＴＥ命令により表示開始位置が指定されている場合は、その位置から入力ガイダンス、あるいは？が表示されます。
（注）ＩＮＰＵＴ命令による入力時のエラーは CLS で解除して、正しいデータを入れてください。

ファイル関連命令

ＩＮＰＵＴ＃…インプット・クロスハッチ　［プログラム］
省略形……Ｉ．＃

［機能］ファイルのデータを、指定した変数に代入します。
［書式］ＩＮＰＵＴ　＃ファイル番号，変数［，変数…］
［参照］ＯＰＥＮ、ＰＲＩＮＴ＃
［説明］
● ＳＩＯ（シリアル入出力装置）に送られてくるデータ、またはラムデータファイル（データファイル）に記録されているデータを指定されている変数に代入します。
● この命令が有効になるのは、ＯＰＥＮ命令で“ＣＯＭ：”，“ＣＯＭ１：”を指定しているとき、または“Ｅ：”のＩＮＰＵＴを指定してオープンしているときだけです。
● ＯＰＥＮ命令で、“ＣＯＭ：”，“ＣＯＭ１：”を指定しているときは、ファイル番号を１に指定します。“Ｅ：”を指定しているときは、ＯＰＥＮ命令で指定したファイル番号（２または３）を指定します。
● ＩＮＰＵＴ＃１，Ｂ（＊）とすると、配列全体の指定になります。文字配列変数のときはＣ＄（＊）のように指定します。くわしくはＰＲＩＮＴ＃命令を参照してください。
● 指定した変数の数よりもファイルのデータが少ない場合は、エラー87になります。

変数への読み込み規則
● 数値
・データの区切りは、コンマ（，）およびスペース、ＣＲ（＆Ｈ０Ｄ）、ＬＦ（＆Ｈ０Ａ）コードです。
・スペースでない最初の文字をデータの始まりとします。最初のスペースは無視します。
・数値化できないデータを読み込んだ場合は０とします。
● 文字
・データの区切りはコンマ（，）およびＣＲ（＆Ｈ０Ｄ）、ＬＦ（＆Ｈ０Ａ）コードです。256文字目を読み込んだときも区切りになります。
・スペースでない最初の文字をデータの始まりとします。最初のスペースは無視します。
・データの始まりがダブルクォーテーション（"）のときは、次のダブルクォーテーションまでを１つのデータとします。
● ＥＯＦ（エンドオブファイル）コードの処理（174ページで指定したコードです。）
・データの前にＥＯＦコードがある場合はエラーになります。
・データの途中にＥＯＦコードがある場合はデータの区切りと見なします。

ファイル関連命令

ＫＩＬＬ………キル　［マニュアル］
省略形……Ｋ．

［機能］プログラムファイルエリアに登録されているファイルを消去します。
［書式］ＫＩＬＬ"ファイル名"⏎
［説明］
● プログラムファイルエリア内の指定したファイルを消去します。拡張子が"．ＢＡＳ"のときは拡張子の指定は省略できます。
● 指定したファイルが存在しないときはエラー94になります。

基本命令

ＬＣＯＰＹ……ラインコピー　［マニュアル］
省略形……ＬＣ．

［機能］プログラム行を複写します。（ＰＲＯモードのマニュアル操作でのみ有効）
［書式］ＬＣＯＰＹ　コピー元開始行番号，コピー元終了行番号，コピー先開始行番号
［参照］ＰＡＳＳ、ＬＩＳＴ
［説明］
● コピー元開始行番号からコピー元終了行番号までのすべてのプログラム行を、コピー先開始行番号から複写します。
● ＧＯＴＯ、ＧＯＳＵＢ、ＲＥＳＴＯＲＥ命令などで参照している行番号は、コピー先のプログラムにおいても元のままです。必要に応じて変更してください。
● パスワードが設定されているときは、ＬＣＯＰＹ命令は無視されます。
● 次の場合はエラーになります。
・指定した行番号が存在しないとき
・コピー元開始行番号がコピー元終了行番号よりも大きいとき
・コピー先開始行がすでに存在するとき
・コピーしたプログラムがすでに存在するプログラムと行番号が混在するとき
・コピーした行番号が65279を超えるとき
・フリーエリアが足りなくなったとき
〈例〉　ＬＣＯＰＹ　１０，１００，２００　10行から100行までのプログラムを200行から同じ増分で複写します。

関数

ＬＥＦＴ＄……レフトドル

省略形……ＬＥＦ.

プログラム　マニュアル

機能　文字列の左側から指定した文字数分を取り出します。

書式　ＬＥＦＴ＄（文字列，式）

参照　ＭＩＤ＄、ＲＩＧＨＴ＄

説明
- 指定された文字列の左から、式の値で指定された桁数（文字数）だけ、文字を取り出します。たとえば、Ａ＄＝〃ＡＢＣＤＥ〃のときＬＥＦＴ＄（Ａ＄，３）はＡ＄の文字列の左側３文字、すなわち“ＡＢＣ”を取り出しなさいという意味になります。
- 式の値は、0～255の範囲の整数でなければなりません。

関数

ＬＥＮ…………レングス

省略形……なし

プログラム　マニュアル

機能　文字列の文字数を求めます。

書式　ＬＥＮ　文字列

説明
- １つの文字列の中に含まれる文字の数（記号、スペース、数字も含みます）を求めます。たとえば、Ａ＝ＬＥＮ〃ＡＢＣ１２３４ﾊﾞﾝ〃とすれば、文字数10が変数Ａに代入されます。(だく点や半だく点も１文字と数えられます。)
- 文字列としては、ＡＢ＄のように文字変数でも指定できます。

基本命令

ＬＥＴ…………レット

省略形……ＬＥ.

プログラム

機能　変数に数値や文字を代入するための命令です。

書式　［ＬＥＴ］変数＝データ［，変数＝データ］…

説明
- 代入文はＬＥＴ命令に続いて代入式を書きます。なお、ＬＥＴ命令は省略できます。
  代入式はＡ＝５＋３、Ｂ＄＝〃ＡＢＣ〃のように、左辺に変数を、右辺に式や文字列を書きます。
  この場合の“＝”は“等しい”という意味ではなく、“左辺の変数に、右辺の内容あるいは計算結果を入れなさい”という意味です。
- 変数とデータの型は同じ（文字型どうし、または数値型どうし）でなければいけません。

ファイル関連命令

ＬＦＩＬＥＳ…エルファイルズ

省略形……ＬＦ.

マニュアル

機能　プログラムファイルエリアに登録されているファイルのファイル名を印字します。

書式　ＬＦＩＬＥＳ ↵

参照　ＦＩＬＥＳ

説明
- プログラムファイルエリアに登録されているすべてのファイル名をプリンタで印字します。

グラフィック命令

ＬＩＮＥ…………ライン

省略形……ＬＩＮ.

プログラム　マニュアル

機能　指定された２点間を線で結びます。

書式

ＬＩＮＥ［(式$_1$，式$_2$)］－（式$_3$，式$_4$）［，{Ｓ／Ｒ／Ｘ}］［，式$_5$］［，{Ｂ／ＢＦ}］

〔Ａ〕　〔Ｂ〕　〔Ｃ〕　〔Ｄ〕　〔Ｅ〕項

参照　ＧＣＵＲＳＯＲ、ＰＳＥＴ

説明
- （式$_1$，式$_2$）で指定される点と（式$_3$，式$_4$）で指定される点を線で結びます。
  《例》　ＬＩＮＥ（0，0）－（１４３，４７）
- 式$_1$～式$_4$の値の範囲は－32768～32767ですが、画面に表示できる範囲は次のとおりです。
  式$_1$、式$_3$：0～143　　画面の左上が（0，0）で、右下が（１４３，４７）です。
  式$_2$、式$_4$：0～ 47
  画面外の領域を指定した場合でも、－32768～32767の範囲内であればエラーにならず、画面内に当たる部分のみが描かれます。
- （式$_1$，式$_2$）を省略した場合は、（0，0）位置、または直前に実行されたＬＩＮＥ命令の（式$_3$，式$_4$）で指定された位置から線が描かれます。
  《例》
```
10  CLS
20  LINE（10，0）－（143，24）
30  WAIT：LINE－（71，47）
```
- 〔Ｃ〕項のＳ、Ｒ、Ｘにより、線に当たるドットを点灯させるか、消すか、あるいは反転させるかを指定します。
  Ｓ……線を描くとき、ドットを点灯させて線を描きます。(ドットをセット)
  Ｒ……線を描くとき、ドットを消灯させて（消して）線を描きます。(ドットをリセット)
  　　線の回りのドットが点灯している場合に線を描くときや、描かれている線を消すときなどに用います。
  Ｘ……線を描くとき、線に当たるドットが点灯しているときは消して、消えているときは点灯させます。(ドットを反転)
  指定を省略した場合は、Ｓを指定したときと同じになります。
- 〔Ｄ〕項の式$_5$の値により、線の種類を指定します。
  たとえば、式$_5$の値が5503（＆Ｈ157Ｆ）の場合、次のような線が描かれます。

16ドット　　左と同じ形が繰り返されて線が描かれます。

この5503（＆Ｈ157Ｆ）を16桁の2進数で表せば、次のようになります。

０１１１１１１１０００１０１０１（７Ｆ１５の順になります。）

上記の図で示した16ドット分と、この２進数を比べてみると、各桁の１に相当するドットが点灯し、０に相当するドットが消えていることがわかります。

このように、式$_5$の値を16桁の２進数に変換したときの各桁が０か１かによって、線の種類が指定されます。

したがって、式$_5$ の値が０のときは線が画面にあらわれず、65535（＆ＨＦＦＦＦ）のときは実線になります。また、式$_5$ を省略した場合も実線になります。
ただし、〔Ｃ〕項でＲが指定されているときは、各桁の１に相当するドットをリセット（消去）し、Ｘが指定されているときは、各桁の１に相当するドットの反転を行います。

- 式$_5$ の値は０～65535（＆ＨＦＦＦＦ）の範囲で指定できます。
- 〔Ｅ〕項のＢ、ＢＦにより、（式$_1$，式$_2$）の点と（式$_3$，式$_4$）の点を結ぶ線を対角線とした長方形を描きます。
  Ｂ………長方形を描く
  ＢＦ……長方形を線で塗りつぶすように描く

（注）
画面はドットで構成されていますので、斜めの線などは正確な直線にならない場合があります。また、曲線も正確な曲線にはなりません。

〈例〉
```
10 CLS :WAIT 0
20 AA$="102812FD122810"
30 GCURSOR(64,28)
40 GPRINT AA$;AA$;AA$
50 LINE(24,0)-(124,47),&HF18F,B
60 LINE(34,3)-(114,44),X,BF
70 GOTO 60
```

基本命令

ＬＩＳＴ………リスト
省略形……Ｌ.　　マニュアル

機能　記憶されているプログラムを表示させます。（ＰＲＯモードのマニュアル操作でのみ有効）

書式　（１）ＬＩＳＴ ↵
（２）ＬＩＳＴ　行番号 ↵
（３）ＬＩＳＴ　ラベル ↵

参照　ＬＬＩＳＴ、ＰＡＳＳ

説明
- 書式（１）では、プログラムの先頭行から、表示できる範囲で表示します。
- 書式（２）では、指定した行番号の行から、表示できる範囲で表示します。指定した番号の行がない場合は、それよりも大きく、かつ一番近い行から表示します。
- 書式（３）では、指定したラベルの書かれている行から、表示できる範囲で表示します。ラベルは163ページを参照ください。
- プログラムが記憶されていないときや、パスワードが設定されているときは、ＬＩＳＴ命令は無視されます。
- プログラム内にないラベルや、プログラムの最終行よりも大きい行番号を指定した場合はエラー40になります。

プリンタ命令

ＬＬＩＳＴ……ラインリスト
省略形……ＬＬ.　　マニュアル

機能　プログラムをプリンタで印字します。（ＰＲＯおよびＲＵＮモードのマニュアル操作でのみ有効）

書式　（１）ＬＬＩＳＴ ↵
（２）ＬＬＩＳＴ ｛行番号／ラベル｝ ↵
（３）ＬＬＩＳＴ［開始行］－［終了行］↵

参照　ＬＩＳＴ、ＰＡＳＳ

説明
- 書式（１）では、計算機内のプログラムをすべてプリンタで印字します。
- 書式（２）では、指定した行番号または指定したラベルのついている行だけを印字します。
- 書式（３）では、指定した開始行から終了行までのプログラムを印字します。（開始行および終了行はラベルも可）
  なお、書式（３）では開始行または終了行の指定を省略できます。（同時に両方を省略することはできません。）
  開始行を省略した場合は、プログラムの先頭行から、指定した終了行までのプログラムを印字します。
  終了行を省略した場合は、指定した開始行から最終行までのプログラムを印字します。
  〈例〉　ＬＬＩＳＴ －２００ ↵　　先頭行から200行までを印字します。
  　　　　ＬＬＩＳＴ １００－ ↵　　100行から最後の行までを印字します。
- 指定した番号の行がない場合は、それぞれの値よりも大きくかつ最も近い行が指定されます。ただし、開始行が終了行よりも大きくなるような指定をするとエラー44になります。
- パスワードが設定されているときは、ＬＬＩＳＴ命令は無視されます。

ファイル関連命令

ＬＮＩＮＰＵＴ＃…ラインインプット・クロスハッチ　　プログラム
省略形……ＬＮＩ.＃

機能　ファイルの１行（255バイト以内）単位のデータを、指定した文字変数に代入します。

書式　ＬＮＩＮＰＵＴ ＃ファイル番号，文字変数［，文字変数……］

参照　ＯＰＥＮ

説明
- ＳＩＯ（シリアル入出力装置）に送られてくるデータ、またはラムデータファイル（データファイル）に記録されているデータを指定されている文字変数に代入します。
- この命令が有効になるのは、ＯＰＥＮ命令で“ＣＯＭ：”，“ＣＯＭ１：”を指定しているとき、または“Ｅ：”のＩＮＰＵＴを指定してオープンしているときだけです。
- ＯＰＥＮ命令で、“ＣＯＭ：”，“ＣＯＭ１：”を指定しているときは、ファイル番号を１に指定します。“Ｅ：”を指定しているときは、ＯＰＥＮ命令で指定したファイル番号（２または３）を指定します。

変数への読み込み規則
- データの区切りは，ＣＲ（＆Ｈ０Ｄ）＋ＬＦ（＆Ｈ０Ａ）コードです。
  256文字目を読み込んだときも区切りになります。ＳＩＯに送られてくるデータの区切りはシリアル入出力の条件設定に従います。
  ＣＲとＬＦコードは、文字変数には代入されません。
- ＥＯＦ（エンドオブファイル）コードの処理
  ・データの前にＥＯＦコードがあるときは、エラーになります。
  ・データの途中にＥＯＦコードがあるときは、ＥＯＦコードを読み込んだ時点で、通信が終了します。

ファイル関連命令

ＬＯＡＤ………ロード
省略形……ＬＯ.　　マニュアル

機能　プログラムファイルエリアに登録されているＢＡＳＩＣプログラムを呼び出します。

書式　ＬＯＡＤ"ファイル名" ↵

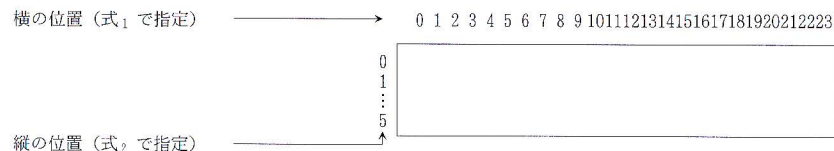
参照　ＳＡＶＥ

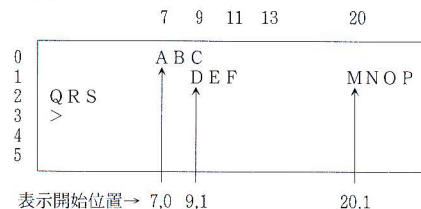
説明　● プログラムファイルエリアから、指定したファイル名のプログラムを呼び出します。
● 拡張子が「．ＢＡＳ」のときのみ、拡張子の記述は省略できます。
● 読み込み終了時、ＳＩＯがオープンしているとクローズされます。
● テキスト（ＴＥＸＴ）プログラムを呼び出そうとするとエラー96になります。

基本命令

ＬＯＣＡＴＥ…ロケート　プログラム
省略形……ＬＯＣ．

機能　表示の開始位置（ポジション）を指定します。
書式　ＬＯＣＡＴＥ　式$_1$〔，式$_2$〕
参照　ＣＬＳ、ＩＮＰＵＴ、ＰＲＩＮＴ
説明　● ＰＲＩＮＴ命令などで表示される内容の表示開始位置（カーソルの位置）を指定します。
● 表示位置は、次の図のようになります。

横の位置（式$_1$で指定）→　0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
縦の位置（式$_2$で指定）→　0 1 ⋮ 5

このように、表示部を横と縦に分け、式$_1$の値で横の位置を指定し、式$_2$の値で縦の位置を指定します。
● 式$_1$の値は０～23、式$_2$の値は０～５の範囲で指定します。この範囲外ではエラー33になります。
● 式$_2$を省略した場合、縦の位置は現在カーソルがある位置になります。

〈例〉
```
 5 CLS
10 LOCATE 7,0:PRINT"ABC"
20 LOCATE 9,1:PRINT"DEF"
30 LOCATE 20,1:PRINT"MNOPQRS"
```
プログラムを実行すると、次のように表示されます。

● ＬＯＣＡＴＥ命令で表示開始位置を指定した場合は、表示の一部分だけを変えることもでき、応用範囲も広がります。スペースに変えると、一部分だけを消したことになります。
● ＬＯＣＡＴＥ命令による指定はＩＮＰＵＴ命令に対しても働きます。

基本命令

ＬＯＦ……エル・オー・エフ　プログラム　マニュアル
省略形……なし

機能　指定したラムデータファイルの未使用領域の大きさを求めます。
書式　ＬＯＦ　ファイル番号
説明　● ファイル番号で指定したラムデータファイルの未使用バイト数を求めます。
● 指定したファイル番号のファイルがＯＰＥＮされていないとエラー85になります。

プリンタ命令

ＬＰＲＩＮＴ…ラインプリント　プログラム　マニュアル
省略形……ＬＰ．

機能　指定した内容をプリンタで印字します。
書式　ＬＰＲＩＮＴ〔｛式／文字列｝〔｛，／；｝｛式／文字列｝……〕〔；〕〕
参照　ＰＲＩＮＴ、ＵＳＩＮＧ
説明　● 項目を１つだけ指定したときは、式の値は紙の右側に詰めて印字し、文字は紙の左端から印字します。文字列が24桁を超える場合は自動的に改行して印字します。
● コンマ（，）を入れて２つ以上の項目を記述すると、１行の印字桁数24桁を左右12桁に分けて印字します。このときも12桁の範囲内で数値は右詰め、文字は左詰めにします。なお、印字内容が12桁を超える場合、数値は仮数部の下位桁を切り捨てて12桁以内で印字し、文字は先頭から12桁を印字します。
● 区切りにセミコロン（；）を使用している場合は紙の左端から連続的に印字します。印字内容が24桁を超える場合は自動的に改行されます。
● プログラム中で末尾がセミコロン（；）の場合は、印字が終了しても改行を行わず、次のＬＰＲＩＮＴ命令で、指定されている内容を前の内容に続けて印字します。
● ＬＰＲＩＮＴのみで、印字する内容が指定されていないときは改行を行います。

関数

ＭＩＤ＄………ミッドドル　プログラム　マニュアル
省略形……ＭＩ．

機能　文字列の中から指定した文字数分を取り出します。
書式　ＭＩＤ＄（文字列，式$_1$，式$_2$）
　式$_1$：文字列の左何文字目から取り出すかを指定します。
　式$_2$：何文字分を取り出すか指定します。
参照　ＬＥＦＴ＄、ＲＩＧＨＴ＄
説明　● 式$_1$は１～255の範囲で指定できます。
● 式$_2$は０～255の範囲で指定できます。ただし、０を指定した場合、文字は得られずNullになります。

〈例〉
```
10 A$="ABCDE"
20 B$=MID$(A$,2,3)
30 PRINT B$
```
｛Ａ＄の文字列の左側２文字目から３文字、つまりＢＣＤを取り出し、Ｂ＄に代入します。

MON…………モニタ
省略形……MO.
マニュアル

機能　機械語モニタモードにします。（RUNおよびPROモードのマニュアル操作でのみ有効）
書式　MON ↵
説明　機械語モニタモードにします。274ページ　「機械語モニタ機能」を参照ください。

基本命令

NEW…………ニュー
省略形……なし
マニュアル

機能　プログラムとデータを消去します。（PROモードのマニュアル操作でのみ有効）
書式　NEW ↵
参照　CLEAR、PASS
説明
- プログラム・データエリア内のプログラム（BASICプログラム）や配列変数、単純変数がすべて消去され、固定変数の内容も消去されます。
- オープンしているファイル（デバイス）をクローズします。
- パスワードが設定されているときはNEW命令が無視されます。

関数

NOT…………ノット
省略形……NO.
プログラム
マニュアル

機能　与えられた数値の否定を取ります。
書式　NOT　式
参照　AND、OR
説明
- ２進数において、否定は次の値を取ります。
  NOT　１＝０
  NOT　０＝１
- 10進数の否定を取った場合は、その10進数を２進数に変換して各桁の否定を取り、その結果を10進数に変換します。
  このときの10進数をXとしたとき、Xとその否定（NOT　X）の間には次の関係があります。
  NOT X＝－（X＋1）
  この関係式から
  NOT　０＝－１
  NOT－１＝０
  NOT－２＝１
  となります。

基本命令

ON～GOTO……オン～ゴートゥー
ON～GOSUB…オン～ゴーサブ
省略形……O. G.、O. GOS.
プログラム

機能　式の値により、指定された行を選択して実行を移します。
書式　ON　式 {GOTO / GOSUB} 行番号$_1$［，行番号$_2$］［，行番号$_3$］……
参照　GOTO、GOSUB
説明
- ONに続く“式”の値が１であれば“行番号$_1$”、２であれば“行番号$_2$”、３であれば“行番号$_3$”というように、“式”の値により指定されている“行番号”が決定され、GOTOやGOSUBの各機能を実行します。
- “式”の値は整数部のみが有効になります。
- “式”の値が１より小さいときや指定されている“行番号”の個数より大きいときは、本命令の次の命令へ実行が移ります。
- “行番号”はラベルを指定することもできます。（ラベルについては163ページ参照）

〈例〉
```
 10 CLS
 20 LOCATE 5,0
 30 INPUT"ハ゛ンコ゛ウ(1-3)?",N
 40 LOCATE 10,2
 50 ON N GOTO 100,200,300
 60 CLS:GOTO 20
 70 END
100 PRINT"FIRST "
110 GOTO 20
200 PRINT"SECOND"
210 GOTO 20
300 PRINT"THIRD "
310 GOTO 20
```

ファイル関連命令

OPEN………オープン
省略形……OP.
プログラム
マニュアル

機能　SIO（シリアル入出力装置）に対するデータの入出力、ミニI／Oへの出力、ラムデータファイルに対するデータの入出力を可能にします。
書式
（1）OPEN"COM:"
（2）OPEN"COM1:"
（3）OPEN"PIO:"
（4）OPEN"E:ファイル名"　FOR　モード　AS　#ファイル番号
（5）OPEN"LPRT:"
参照　CLOSE
説明
- 書式（1）、（2）、（3）では、SIOに対する入出力を可能にします。（SIO回路をオープンします。）ファイル名を書くことや、入出力の指定はできません。
  書式（1）は半二重通信、書式（2）は全二重通信、書式（3）は８ビットの入出力指定（371ページ参照）です。
- 書式（1）、（2）の入出力条件はTEXTモードのSIOで設定します。

● 書式（4）では、ラムデータファイル（シーケンシャルデータ）へのデータの入出力を可能にします。なお、ＯＰＥＮ命令実行前に、ラムデータファイルモードでInit機能を使って、ファイルの確保と容量の指定を行っていなければなりません。(181ページ参照)
ファイルが確保されていないとエラーになります。

● 書式（4）で指定するモードは次のとおりで、ファイル番号は2または3を指定します。
ＩＮＰＵＴ……データの読み出しを行います。
ＯＵＴＰＵＴ…データの書き込みを行います。すでにデータがある場合は、データの書き替えになります。
ＡＰＰＥＮＤ…データの追加書き込みを行います。

● 書式（5）では、ミニＩ／Ｏに対する出力を可能にします。(367ページ参照)

● 書式（4）以外で複数の回路（ファイル）を同時にオープンしておくことはできません。(すべて自動的にファイル番号1を使用します。)
どれか1つがオープンしているときにＯＰＥＮ命令を実行するとエラー86になります。
ただし、これとは別に、書式（4）では同時に2つのファイルをオープンしておくことができます。

関数

ＯＲ……………オア
省略形……なし
プログラム
マニュアル

機能　式と式との論理和を計算します。また、条件式の結合を行います。

書式　式　ＯＲ　式
条件式　ＯＲ　条件式

参照　ＡＮＤ、ＮＯＴ、ＩＦ

説明
● 2進法において、論理和は次のような値を取ります。
1　ＯＲ　1＝1　　0　ＯＲ　1＝1
1　ＯＲ　0＝1　　0　ＯＲ　0＝0

● 10進数の論理和を求めた場合は、10進数を2進数に変換したうえで、各桁の論理和を求め、その結果を10進数に戻します。
たとえば、41と27の論理和は次のように計算されます。
41　ＯＲ　27＝59

```
    101001………41
OR〈
    011011………27
    ─────────────
    111011………59
```

41と27をそれぞれ2進数に変換し、各桁のＯＲを取ります。そして、その結果を10進数に変換すれば59になります。

● 2つ以上の条件のうち、いずれかを満足するような条件を1つの式で表します。
〈例〉　Ａ＜0　ＯＲ　Ａ＞6　　Ａは0よりも小さいか、あるいは6よりも大きい。
ＩＦ　Ａ＝1　ＯＲ　Ｂ＝1　ＯＲ　Ｃ＝1　ＴＨＥＮ…
Ａ、Ｂ、Ｃのいずれかが1のとき、ＴＨＥＮに続く命令を実行します。

グラフィック命令

ＰＡＩＮＴ………ペイント
省略形………ＰＡＩ.
プログラム
マニュアル

機能　指定した点を囲む領域を、指定した模様で塗りつぶします。

書式　ＰＡＩＮＴ（式$_1$，式$_2$），式$_3$

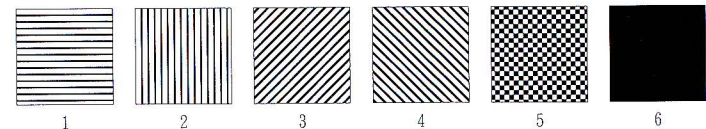
参照　ＣＩＲＣＬＥ、ＧＣＵＲＳＯＲ、ＬＩＮＥ

説明
● （式$_1$，式$_2$）で指定した点を囲む領域を、式$_3$で指定した模様で塗りつぶします。
なお、ＣＩＲＣＬＥ命令で円の内部を塗りつぶしたときの模様や表示している文字も境界になります。

● 式$_1$、式$_2$で指定できる範囲は−32768～32767です。画面内は、式$_1$は0～143、式$_2$は0～47の範囲です。

● 画面外の点を指定したときはＰＡＩＮＴ命令は無視されます。

● ＰＡＩＮＴ命令を実行するためには、ワーク用として約1440バイト以上のフリーエリアが必要です。

● 式$_3$で塗りつぶす模様を次のように指定します。

基本命令

ＰＡＳＳ………パス
省略形……ＰＡ.
マニュアル

機能　パスワードの設定あるいは解除を行います。（ＰＲＯおよびＲＵＮモードのマニュアル操作でのみ有効）

書式　ＰＡＳＳ〃文字〃⏎

参照　ＮＥＷ、ＳＡＶＥ、ＢＬＯＡＤ、ＢＳＡＶＥ

説明
● 作成したプログラムを他の人に知られたくないときや変更されたくないときに、ある言葉や記号をパスワード（暗号）とし、パスワードを与えないかぎり、計算機内のプログラムを呼び出せないようにできます。(プログラムの秘密化)

● パスワードは、8文字までの英文字、カナ文字、数字、記号を使用することができます。〃〃（Null）はパスワードとして設定できません。8文字以上のパスワードを宣言したときは、頭から8文字のみが有効となり、設定または解除が行われます。

● パスワードが宣言されていない状態のとき、ＰＡＳＳ命令を実行すれば、そのとき計算機内にあるＢＡＳＩＣプログラムに対してパスワードが設定され、秘密プログラムになります。

● 秘密化されたプログラムに対しては、ＬＩＳＴ命令、▲や▼など、プログラムの呼び出しにかかわる命令や機能、行の追加・削除などの機能は働きません。

● 秘密プログラムはＮＥＷ、ＤＥＬＥＴＥ命令でも消去されず、保護されます。ＡＵＴＯ、ＲＥＮＵＭ、ＬＣＯＰＹ、ＬＬＩＳＴ命令は無視されます。また、プログラムファイルエリアへ登録したりポケコンや他の出力機器に出力することもできません。

● 秘密プログラムを解除する場合はもう一度同じパスワードを宣言します。(パスワードが違っているとエラー92になります。)

● 計算機内にＢＡＳＩＣプログラムが入っていないときにＰＡＳＳ命令を実行するとエラー14になり、パスワードは設定されません。

関数

ＰＥＥＫ………ピーク　プログラム　マニュアル
省略形………ＰＥ.

機能　機械語プログラムやデータを直接読み出します。
書式　ＰＥＥＫ　番地
参照　ＰＯＫＥ、ＣＡＬＬ
説明
- 指定した番地からデータを読み出します。
- 番地は０～65535（＆Ｈ０～＆ＨＦＦＦＦ）の値で指定します。
- 読み出されるデータは、０～255（＆Ｈ０～＆ＨＦＦ）の値になります。
- パスワードが設定されているとき、本命令をマニュアルで実行するとエラー93になります。

〈例〉　4001番地（16進表記）のデータを読み出し、変数Ａに入れます。

```
Ａ＝ＰＥＥＫ　＆Ｈ４００１
```

グラフィック命令

ＰＯＩＮＴ………ポイント　プログラム　マニュアル
省略形……ＰＯＩ.

機能　指定したドットの状態を読みとる命令です。
書式　ＰＯＩＮＴ（式$_1$，式$_2$）
参照　ＧＣＵＲＳＯＲ、ＰＳＥＴ、ＰＲＥＳＥＴ
説明
- （式$_1$，式$_2$）で指定されたドットが点灯しているときは１、消えているときは０が得られます。指定されたドットが画面の範囲外にある場合は－１が得られます。
- 式$_1$、式$_2$の値は－32768～32767の範囲内で指定できます。ただし、画面内は式$_1$が０～143、式$_2$が０～47の範囲になります。

〈例〉

```
１０　ＣＬＳ　：ＷＡＩＴ　３：Ａ＝５０
２０　ＬＩＮＥ（２０，８）－（２０，３９）　← 画面に縦の線を２本描きます
３０　ＬＩＮＥ（１００，８）－（１００，３９）
４０　ＰＳＥＴ（Ａ，２４）　　　　← ２本の線の間にドット（点）を点灯させます
５０　Ｂ＝ＰＯＩＮＴ（Ａ＋１，２４）　← 右側のドットが点灯しているかどうかを調べます
６０　ＩＦ　Ｂ　ＴＨＥＮ　１５０　　← もし、点灯していたら150行へ行きます
７０　ＰＳＥＴ（Ａ＋１，２４）　　← 消えているときは、そのドットを点灯させます
８０　ＰＲＥＳＥＴ（Ａ，２４）　　← そして、前に点灯していたドットを消します
９０　Ａ＝Ａ＋１　　　　　　　　← 横方向の位置を右へ寄せます
１００　ＧＯＴＯ　５０　　　　　← 50行に戻ります
１５０　Ｂ＝ＰＯＩＮＴ（Ａ－１，２４）　← 左が点灯しているかどうかを調べます
１６０　ＩＦ　Ｂ　ＴＨＥＮ　５０　　← もし、点灯していたら50行へ行きます
１７０　ＰＳＥＴ（Ａ－１，２４）　　← 消えているときは、そのドットを点灯させます
１８０　ＰＲＥＳＥＴ（Ａ，２４）　　← そして、前に点灯していたドットを消します
１９０　Ａ＝Ａ－１　　　　　　　　← 横方向の位置を左に寄せます
２００　ＧＯＴＯ　１５０　　　　　← 150行へ戻ります
```


このプログラムを実行すると、画面に描かれた２本の線の間をドットが行ったりきたりします。

基本命令

ＰＯＫＥ………ポーク　プログラム　マニュアル
省略形……ＰＯＫ.

機能　機械語プログラムやデータをメモリに直接書き込みます。
書式　ＰＯＫＥ　番地，データ１，データ２，……
参照　ＣＡＬＬ、ＰＥＥＫ
説明
- 指定した番地をデータ記憶の開始番地として、データ１、データ２……と、順にメモリに記憶していきます。
- 番地は０～65535（＆Ｈ０～＆ＨＦＦＦＦ）の値で指定します。
- データは１バイト単位で指定します。したがって、各データの範囲は０～255（＆Ｈ００～＆ＨＦＦ）です。
- パスワードが設定されているとき、本命令をマニュアルで実行するとエラー93になります。

〈例〉　＆Ｈ０１、＆Ｈ０２、＆Ｈ０３を7000～7002番地（16進表記）に書き込みます。

```
ＰＯＫＥ＆Ｈ７０００，＆Ｈ０１，＆Ｈ０２，＆Ｈ０３
```

ご注意

この命令を誤って使用するとＢＡＳＩＣプログラムやシステムエリアを破壊し、異常が発生することがあります。

グラフィック命令

ＰＲＥＳＥＴ………ポイント・リセット　プログラム　マニュアル
省略形……ＰＲＥ.

機能　画面上の指定されたドット（点）を消します。
書式　ＰＲＥＳＥＴ（式$_1$，式$_2$）
参照　ＰＳＥＴ、ＧＣＵＲＳＯＲ、ＰＯＩＮＴ
説明
- （式$_1$，式$_2$）で指定されたドットを消します。
- 式$_1$、式$_2$の値は－32768～32767の範囲内で指定できます。ただし、画面内は式$_1$が０～143、式$_2$が０～47の範囲になります。

〈例〉

```
１０　ＣＬＳ　：ＷＡＩＴ　０
２０　ＬＩＮＥ（０，０）－（１４３，４７），ＢＦ
３０　ＦＯＲ　Ｉ＝－１　ＴＯ　１　ＳＴＥＰ　２
４０　ＦＯＲ　Ｘ＝－２４　ＴＯ　２４　ＳＴＥＰ　０.５
５０　Ｙ＝Ｉ＊ＳＱＲ　ＡＢＳ（２４＊２４－Ｘ＊Ｘ）
６０　ＰＲＥＳＥＴ（Ｘ＋７１，Ｙ＋２４）
```

```
70　NEXT　X:NEXT　I
80　WAIT　:GPRINT
```

このプログラムを実行すると、塗りつぶされた四角形の中に円が描かれます。

基本命令

ＰＲＩＮＴ……プリント

省略形……Ｐ.

プログラム　マニュアル

機能 指定した内容を表示部に表示します。

書式 ＰＲＩＮＴ［｛式／文字列｝［｛，／；｝｛式／文字列｝………］［；］］

参照 ＬＯＣＡＴＥ、ＬＰＲＩＮＴ、ＵＳＩＮＧ、ＷＡＩＴ

説明
- 項目を１つだけ指定したときは、式の値は表示部の右側に詰めて表示し、文字は表示部の左端から表示します。

〈例〉

```
10　PRINT　"ABCD"
20　PRINT　123
```

ただし、ＬＯＣＡＴＥ命令により、表示開始位置が指定されているときは、その位置から表示します。

- コンマ（,）で区切って２つ以上の項目を指定したときは、表示部を12桁ずつに区切り、最初に指定されている内容から順番に表示していきます。この場合も、12桁の範囲内で式の値は右側に詰めて表示し、文字は左側から表示します。なお、数値または文字が12桁を超える場合は次のように処理されます。

①数値が12桁を超える場合（指数表示において、仮数部が７桁以上になる場合）は、仮数部の下位桁が切り捨てられます。

②文字が12桁を超える場合は、先頭から12桁のみを表示します。

〈例〉

```
10　CLS
20　A=123:B=5/9:C$=
　　"ABCD"
30　PRINT　"A=",A
40　PRINT　A,C$,B
```

```
A=　　　　　　　　　 123.
　　　　　123.ABCD
 5.55555E-01
```

↑仮数部の下位桁を切り捨て

- 区切りにセミコロン（；）を使用している場合は、指定された内容を続けて表示します。

〈例〉

```
10　CLS
20　A=123:B=5/9:C$=
　　"ABCD"
30　PRINT　"A=";A
40　PRINT　A;C$;B;"VWXYZ"
```

```
A=123.
123.ABCD5.555555556E-01V
WXYZ
```

- 末尾がセミコロン（；）の場合は、その前に指定されている内容を左に詰めて表示し、その表示した内容の最後に続く桁が、次のＰＲＩＮＴ命令に対する表示開始位置となります。

〈例〉

```
10　A=123:B=45
20　CLS:　PRINT
　　"123*45=";
30　C=A*B
```

```
123*45=5535.
```

↑この桁からＣの内容を表示


```
40　PRINT　C
```

- ＰＲＩＮＴのみで、表示する内容が指定されていないときは、改行を行います。
- ＬＯＣＡＴＥ命令や、末尾がセミコロンのＰＲＩＮＴ文で表示開始位置が指定されている場合は、その位置から表示を開始します。

なお、このとき表示する内容の項目が（,）で区切られている場合、最初の項目は12桁の範囲にこだわらずに表示されます。

- １つのＰＲＩＮＴ命令で表示に使用する桁数は、255桁までです。255を超えた桁は切り捨てられます。

◎ＰＲＩＮＴ→ＬＰＲＩＮＴ指定

本機はＰＲＩＮＴ命令を、必要に応じて印字命令に切り替えることができます。
たとえば、計算機本体のみで使用しているときは、ＰＲＩＮＴ命令を表示命令として画面に計算結果を表示させ、プリンタを接続しているときは印字命令として計算結果などをプリントさせることができます。

指定・解除

プリンタが接続されているとき、マニュアルあるいはプログラムで

　ＰＲＩＮＴ＝ＬＰＲＩＮＴ

の命令を実行すると、ＰＲＩＮＴ命令はすべてＬＰＲＩＮＴ命令と同様に働きます。
この機能は

　ＰＲＩＮＴ＝ＰＲＩＮＴ

の命令で解除できます。ＲＵＮ命令の実行、［SHIFT］＋［CLS］(CA) の操作、電源のオフ・オンなどでも解除され、ＰＲＩＮＴ命令は通常の表示命令に戻ります。マニュアルでＰＲＩＮＴ＝ＬＰＲＩＮＴ命令を実行させ、有効に働かせるには、次の方法を用います。

- 命令実行後、ＧＯＴＯ命令でプログラムをスタートさせる。（163ページ参照）
- ＩＮＰＵＴ命令などにより、プログラムがストップしているときに、この命令を実行する。

ファイル関連命令

ＰＲＩＮＴ＃…プリント・クロスハッチ

省略形……Ｐ．＃

プログラム　マニュアル

機能 指定したデータをＳＩＯ（シリアル入出力装置）、ラムデータファイルに出力します。

書式 ＰＲＩＮＴ　＃ファイル番号，データ［｛，／；｝データ…］［｛，／；｝］

参照 ＯＰＥＮ、ＩＮＰＵＴ＃

説明
- 指定したデータをＳＩＯから出力します。またはラムデータファイル（シーケンシャルファイル）に記録します。
- この命令が有効になるのは、ＯＰＥＮ命令で“ＣＯＭ：”、“ＣＯＭ１：”を指定していると き、または“Ｅ：”のＯＵＴＰＵＴあるいはＡＰＰＥＮＤを指定してオープンしているときだけです。
- ＯＰＥＮ命令で、“ＣＯＭ：”，“ＣＯＭ１：”を指定しているときは、ファイル番号を１に指定します。“Ｅ：”を指定しているときは、ＯＰＥＮ命令で指定したファイル番号（２または３）を指定します。
- “データ”は、一般式（数式、文字式）の他に変数を指定できます。

ＰＲＩＮＴ＃１，Ｂ（＊）とすると、配列全体の指定になります。文字配列変数のときはＣ＄（＊）のように指定します。

配列全体の出力順序

〈例〉　一次元配列　Ｂ（０）→Ｂ（１）→Ｂ（２）→……
　　　　二次元配列　Ｃ（０，０）→Ｃ（０，１）→Ｃ（０，２）→……
　　　　　　　　　　Ｃ（１，０）→Ｃ（１，１）→Ｃ（１，２）→……
　　　　　　　　　　　　⋮

● 配列全体を指定したときは、各要素を出力するごとにＣＲ、ＬＦコードを出力します。
たとえば、ＰＲＩＮＴ＃１，Ｂ（＊）はＰＲＩＮＴ＃１，Ｂ（０）：ＰＲＩＮＴ＃１，Ｂ（１）：ＰＲＩＮＴ＃１，Ｂ（２）……と指定した場合と同じ状態になります。

● ″データ″の後ろがコンマ（，）のときは20桁を１ゾーンとして出力します。
データが20桁以内なら１ゾーンの領域の中で、数値は右詰め、文字は左詰めで出力します。文字データが20桁を超えている場合は、必要なゾーン数を確保して出力します。ゾーンの残りの桁はスペースで埋めます。

〈例〉
```
A$="ABC":C=123:D=-5.2E10
PRINT#1,A$,C,D
出力 ABC␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣123.␣  ←出力フォーマットの説明参照（下記）
-5.2E+10␣CRLF
```

● ″データ″の後ろがセミコロン（；）のときは、データを続けて出力します。

〈例〉
```
A$="ABC":C=123:D=-5.2E10
PRINT#1,A$;C;D
出力 ABC␣123.␣-5.2E+10␣CRLF
```

● 指定された″データ″をすべて出力すると、最後にＣＲ（＆Ｈ０Ｄ）とＬＦ（＆Ｈ０Ａ）コードが送られます。ただし、データの最後にコンマ（，）またはセミコロン（；）が指定されているときはＣＲ、ＬＦコードは出力されません。

● 数値データの出力フォーマット
①数値データの後ろには１桁分のスペースがつけられます。
②数値の前には符号桁が１桁つきます。数値が負数の場合は、この桁が－（マイナス符号）になり、正数の場合はスペースになります。
③指数形式で出力される場合は、仮数部の後ろに指数部を示す記号（Ｅ）、符号、数値２桁（１桁の場合は前に０をつける）が出力されます。

〈例〉
```
PRINT#1,123456789871
              ␣1.23456789E+10␣CRLFを出力
```

④未定義の変数（ＣＬＥＡＲ実行後のＡＢなど）を指定した場合は０が出力されます。

● 文字データの出力フォーマット
①指定されているデータをそのままアスキー形式で出力します。
②ＣＨＲ＄（０）は″″（Null：何もない状態）になります。
③未定義の変数（ＣＬＥＡＲ実行後のＡＢ＄など）を指定した場合は″″（Null）が出力されます。したがって、本機やパソコンでは何も表示されません。

● 文字や配列全体以外の文字変数を指定するとき、コンマ（，）やセミコロン（；）で続けるとデータの区切りコードがつかないため、ラムデータファイルから読み込むときなどにデータを分けることができません。したがって、次のように１命令に１データを指定するようにしてください。

```
PRINT#1,"ABC"
PRINT#1,AB$
```


グラフィック命令

ＰＳＥＴ………ポイント・セット

省略形……ＰＳ．

プログラム　マニュアル

機能　画面上の指定されたドット（点）の点灯または反転を行います。

書式　（１）ＰＳＥＴ（式$_1$，式$_2$）
　　　（２）ＰＳＥＴ（式$_1$，式$_2$），Ｘ

参照　ＰＲＥＳＥＴ、ＧＣＵＲＳＯＲ、ＰＯＩＮＴ

説明
● 書式（１）では、（式$_1$，式$_2$）で指定されたドットを点灯させます。
● 書式（２）では、（式$_1$，式$_2$）で指定されたドットが点灯しているときは消し、消えているときは点灯させます。
● 式$_1$、式$_2$の値は－32768～32767の範囲内で指定できます。ただし、画面内は式$_1$が０～143、式$_2$が０～47の範囲になります。

〈例〉
```
10 CLS :WAIT 0:DEGREE
20 FOR A=0 TO 420 STEP 3
30 B=-1*SIN A
40 Y=INT(B*24)+24
50 X=INT(A/6.25)
60 PSET(X,Y)
70 NEXT A
80 WAIT :GPRINT
```

このプログラムを実行すると、サインカーブが描かれます。

基本命令

ＲＡＤＩＡＮ…ラディアン

省略形……ＲＡＤ．

プログラム　マニュアル

機能　角度単位を“ＲＡＤＩＡＮ”（ラディアン）に設定します。

書式　ＲＡＤＩＡＮ

参照　ＤＥＧＲＥＥ、ＧＲＡＤ

説明
● 三角関数、逆三角関数、座標変換で扱う角度の単位を“ＲＡＤＩＡＮ”単位〔rad〕に設定します。
（１直角＝π／２〔rad〕）

基本命令

ＲＡＮＤＯＭＩＺＥ…ランダマイズ

省略形……ＲＡ．

プログラム　マニュアル

機能　ＲＮＤ命令の使用に先立って乱数のタネを植えつけるものです。

書式　ＲＡＮＤＯＭＩＺＥ

参照　ＲＮＤ

説明　● ＲＮＤ命令により乱数を発生させた場合、電源を入れ直すと常に同じ乱数系列を発生します。しかし、電源を入れた後にＲＡＮＤＯＭＩＺＥ命令を実行すれば、実行のたびに違った乱数を発生させます。

基本命令

ＲＥＡＤ………リード　プログラム

省略形……ＲＥＡ.

機能　ＤＡＴＡ文に続いて指定されているデータを変数に読み込みます。

書式　ＲＥＡＤ　変数［，変数］………

参照　ＤＡＴＡ、ＲＥＳＴＯＲＥ

説明　● 変数にデータを代入する方法の１つです。変数はＲＥＡＤ文で指定し、データはＤＡＴＡ文で指定します。

〈例〉

● １つのプログラムの中に何回でもＲＥＡＤ文およびＤＡＴＡ文を書くことができますが、データは何行に分けて書いても一連のデータと見なされ、小さい番号の行のデータから順番に変数に代入されます。また、複数のプログラムが書き込まれ、それぞれのプログラムにＤＡＴＡ文がある場合でも、それは一連のデータと見なされます。
したがって、それぞれのプログラム内のデータを使用するときは、プログラムの最初の行に、ＲＥＳＴＯＲＥ命令を書き込んでください。

● データと変数はその型（数値変数か文字変数かの型）が一致していなければなりません。

● 読み込もうとして、読み込むデータがない場合はエラー53になります。

● ＤＡＴＡ文では文字データを" "で囲まずに指定できますが、データの前後にスペースが指定されていても、スペースはないものと見なされます。

基本命令

ＲＥＭ…………リマーク　プログラム

省略形……なし

機能　プログラムに注釈を入れます。

書式　ＲＥＭ　注釈

説明　● プログラムの実行には関係なく、プログラムリストなどをわかりやすくする目的で、プログラムの先頭や途中に注釈を入れておくためのものです。

● 同じラインで、ＲＥＭの後に書かれている内容はすべて注釈とみなされ、プログラム実行時は無視されます。

● ＲＥＭの代わりに’（シングルクォーテーション）を使用することもできます。

〈例〉
```
１０　ＲＥＭ　キンリ　ケイサン
　：
２００　’サブルーチン
　：
```


基本命令

ＲＥＮＵＭ……リナンバー　マニュアル

省略形……ＲＥＮ.

機能　プログラムの行番号をつけ直します。（ＰＲＯモードのマニュアル操作でのみ有効）

書式　ＲＥＮＵＭ［新行番号］［，［旧行番号］［，増分］］↵

参照　ＰＡＳＳ

説明　● “旧行番号”で指定したプログラムの行番号を“新行番号”に書き換え、それ以降の行番号を、“増分”で指定した値に従って順次書き換えていきます。

● “旧行番号”で指定した行番号がプログラム内に存在しない場合はエラー40になります。

● 新行番号、旧行番号、増分を省略した場合、旧行番号はプログラムの最初の行、新行番号、増分はそれぞれ10が指定されたものとみなされます。

〈例〉　ＲＥＮＵＭ↵
　プログラム全体の行番号を10から10ステップきざみでつけ直します。
　ＲＥＮＵＭ　１００，５０，１０↵
　行番号５０が１００になり、それ以後、行番号を10ステップきざみで最後までつけ直します。

● ＲＥＮＵＭ命令は、ＧＯＴＯ、ＧＯＳＵＢ、ＩＦ～ＴＨＥＮ～ＥＬＳＥ、ＯＮ～ＧＯＴＯ、ＯＮ～ＧＯＳＵＢ、ＲＥＳＴＯＲＥ命令などで参照している行番号も新しい行番号に対応させて自動的につけ直します。
なお、参照する行番号がプログラム内に存在しないときはエラー40になります。

● 上記の命令で参照している行番号またはラベルの中に行番号やラベルでない文字や記号が入っていると、それより後に書かれている行番号のつけ直しは行いません。

〈例〉　ＧＯＴＯ　１０＊２
　ＯＮ　Ａ　ＧＯＴＯ"ＡＢＣ"，２００，ＡＢ，４００
　～で示す部分は行番号やラベルとなりません。＿で示す部分は行番号とみなされ、つけ直しを行います。

● 新しくつけ直す行番号が65279を超える場合はエラー41になります。

● 実行の順番が変わるような指定（たとえば10、20、30の３つの行がある場合、ＲＥＮＵＭ15，30を実行するとき）を行うとエラー43になります。

● 長いプログラムに対してＲＥＮＵＭ命令を実行すると、少し時間がかかります。表示部右端に＊を１つ表示しているときは BREAK ON で中断すると実行前のプログラムの状態に戻ります。＊を２つ表示しているときは BREAK ON は無視されます。

● パスワードが設定されているときは、ＲＥＮＵＭ命令は無視されます。

基本命令

ＲＥＰＥＡＴ～ＵＮＴＩＬ…リピート～アンティル　プログラム

省略形……ＲＥＰ.　ＵＮ.

機能　ＲＥＰＥＡＴとＵＮＴＩＬの間に書かれた命令を指定された条件が満たされるまで、繰り返し実行します。

書式　ＲＥＰＥＡＴ
　実行文１

ＵＮＴＩＬ　条件式
　実行文２

参照　ＦＯＲ～ＮＥＸＴ、ＷＨＩＬＥ～ＷＥＮＤ

説明
● ＲＥＰＥＡＴ以降の命令（実行文１）を実行し、ＵＮＴＩＬ文で条件式の判断をします。
条件が成立した場合は、実行文２に移ります。（繰り返しループの終了）
条件が成立しなかった場合は、実行文１を再実行し、条件が成立するまで繰り返します。（この繰り返し部分をＲＥＰＥＡＴ～ＵＮＴＩＬループと呼びます。）
● ＲＥＰＥＡＴとＵＮＴＩＬは必ず対にして使用します。
● ＲＥＰＥＡＴ～ＵＮＴＩＬループの中に、別のＲＥＰＥＡＴ～ＵＮＴＩＬループを入れることができます。ただし、中に入るループは外のループ内に完全に入っていなければなりません。したがって、ループが交差するような使いかたはできません。（交差している場合はエラーになります。）
この条件でループを最大22段まで重ねて使う（深みをもたせる）ことができます。（379ページのスタック欄を参照）
● ＲＥＰＥＡＴ～ＵＮＴＩＬループの外からループ内に飛び込むことはできません。飛び込ませるとエラーになります。

（注）・ＲＥＰＥＡＴ～ＵＮＴＩＬループから外に飛び出した場合でも、そのループは終了したことにはなりません。プログラムによっては（ＲＥＰＥＡＴ文を何回か実行するようなプログラムの場合）ＲＥＰＥＡＴ～ＵＮＴＩＬの深みエラーが発生することがあります。
・ＲＥＰＥＡＴ～ＵＮＴＩＬループ内ではＣＬＥＡＲ、ＥＲＡＳＥ、ＤＩＭ命令は使用できません。

〈例〉
```
10 A=0:B=0
20 REPEAT
30     A=A+B
40     INPUT  B  ← データ入力
50 UNTIL  B<0  ← 負の値が入力されるまでREPEAT～
60 PRINT  A       UNTILを繰り返し実行します。
70 END
```
このプログラムはデータを入力しその累計を求めます。データとして負の値を入力すると、累計値を表示して終わります。

基本命令

ＲＥＳＴＯＲＥ…リストア　［プログラム］
省略形……ＲＥＳ.

機能　ＲＥＡＤ命令に続く変数に読み込まれるデータの順番を変えます。
書式　ＲＥＳＴＯＲＥ［読み込み開始行］
参照　ＲＥＡＤ、ＤＡＴＡ
説明
● ＲＥＡＤ命令の実行時、ＤＡＴＡ命令で指定されているデータのどのデータを読み込むかは常に計算機に記憶されていますが、この読み込むデータの順番を強制的に変えるときに使用します。
● “読み込み開始行”を指定すると、その行のＤＡＴＡ命令または指定した開始行以降の最も小さい行番号のＤＡＴＡ命令から読み込みを開始します。
● “読み込み開始行”を省略するとプログラムの最初のＤＡＴＡ命令のデータから読み込みを開始します。
● “読み込み開始行”は、行番号またはラベルで指定します。

関数

ＲＩＧＨＴ＄…ライトドル　［プログラム］［マニュアル］
省略形……ＲＩ.

機能　文字列の右側から指定した文字数分を取り出します。
書式　ＲＩＧＨＴ＄（文字列，式）
参照　ＬＥＦＴ＄、ＭＩＤ＄
説明
● 指定された文字列の右から、式の値で指定された桁数（文字数）だけ、文字を取り出します。
● 式の値は０～255の範囲の整数でなければなりません。

〈例〉
```
10 A$="ABCDE"
20 B$=RIGHT$(A$,3) ←A$の文字列の右側3文字すなわちCDEを取
30 PRINT  B$          り出しB$に代入します。
```

関数

ＲＮＤ…………ランダム　［プログラム］［マニュアル］
省略形……ＲＮ.

機能　乱数（疑似乱数）を発生させます。
書式　ＲＮＤ　式
参照　ＲＡＮＤＯＭＩＺＥ
説明
● ＲＮＤ　ＸにおいてＸの値により次のような乱数を得ることができます。
①Ｘが２以上の場合
　Ｘが整数のとき：１からＸの値以下の乱数を発生します。
　（１≦ＲＮＤ　Ｘ≦Ｘ）
　Ｘが小数部を含むとき：１からＸの整数部に１を加えた値以下の乱数を発生します。
　（１≦ＲＮＤ　Ｘ≦（ＩＮＴ　Ｘ）＋１）
　ただし、この場合は小数部の値によって乱数の発生に差が生じます。
②Ｘが負数の場合：同じ乱数（乱数列）を発生させるために、初期値を一定にします。
③Ｘが０から２未満の場合：１未満の乱数を発生します。
　（０＜ＲＮＤ　Ｘ＜１）
● 乱数の有効桁数は10桁です。

〈例〉
```
 10 CLS
 20 USING "##"
 30 FOR A=0 TO 2
 40 Y=RND -1
 50 FOR B=0 TO 3
 60 C= RND 9
 70 PRINT C;
 80 NEXT B
 90 PRINT: NEXT A
100 END
```
40行は同じパターンの乱数を発生させるために入れています。
これにより、同じ乱数列が３回表示されます。
40行を削除してプログラムを実行してみてください。
同じ乱数が発生されなくなります。

プログラムを実行すれば同じ乱数列が表示されます。
```
7  1  4  3 ┐
7  1  4  3 ├ 3回表示されます。
7  1  4  3 ┘
```

基本命令

ＲＵＮ…………ラン　マニュアル

省略形……Ｒ.

機能　プログラムの実行を開始します。（ＲＵＮモードのマニュアル操作でのみ有効）

書式　（１）ＲＵＮ ↵

（２）ＲＵＮ ｛行番号／ラベル｝ ↵

参照　ＧＯＴＯ

説明
- 書式（１）では、プログラムの一番小さい行番号から実行が開始されます。
- 書式（２）では、指定した行あるいはラベルが書かれている行からプログラムの実行が開始されます。（ラベルについては163ページ参照）
- 指定した行番号あるいはラベルがない場合はエラー40になります。
- プログラムに同じラベルが２個以上書かれているときは、行番号の小さいほうが実行されます。
- ＲＵＮ命令によりプログラムの実行が開始されたときの計算機の状態については162ページをご覧ください。

ファイル関連命令

ＳＡＶＥ………セーブ　マニュアル

省略形……ＳＡ.

機能　ＢＡＳＩＣプログラムをプログラムファイルエリアに登録します。

書式　ＳＡＶＥ　〝ファイル名〟

参照　ＬＯＡＤ、ＦＩＬＥＳ

説明
- ＢＡＳＩＣプログラムにファイル名をつけて登録します。
- ファイル名は８文字以下の名前と、拡張子を３文字まで指定できます。
- 拡張子の記述を省略した場合は「．ＢＡＳ」になります。
- 既存のファイル名を指定した場合は、そのファイルの書き換えになります。ただし、既存のファイル名がＴＥＸＴ（テキスト）ファイルの場合はエラー96になります。
- 計算機内のプログラムが秘密化されているときや、プログラムがないときは、ＳＡＶＥ命令は無視されます。

基本命令

ＳＴＯＰ………ストップ　プログラム

省略形……Ｓ.

機能　プログラムの実行を一時停止させます。

書式　ＳＴＯＰ

参照　ＣＯＮＴ

説明
- プログラムにバグ（誤り）が発生したとき、プログラムの実行を途中で止めて、変数の状態を調べたり、変数に数値などを代入したりしてバグを探します。このようなとき、プログラムを止めたい位置にＳＴＯＰ命令を書き込みます。
- この命令が実行されると「ＢＲＥＡＫ　ＩＮ　２００」のように、実行した行番号を伝えるブレークメッセージを表示して停止します。
- この命令により停止したプログラムの実行を再開させるときはＣＯＮＴ命令を使用します。

関数

ＳＴＲ＄………ストリングドル　プログラム　マニュアル

省略形……ＳＴＲ.

機能　数値を文字列に変換します。

書式　ＳＴＲ＄　式

参照　ＶＡＬ

説明
- 数値を文字列の形に変換します。

たとえば、Ｂ＝１２３４のとき

Ａ＄＝ＳＴＲ＄　Ｂ

として実行すれば、Ａ＄には〝１２３４〟という文字列が代入されます。

基本命令

ＳＷＩＴＣＨ～ＣＡＳＥ～ＤＥＦＡＵＬＴ～ＥＮＤＳＷＩＴＣＨ
………スイッチ～ケース～デフォルト～エンドスイッチ　プログラム

省略形……ＳＷ.　ＣＡＳ.　ＤＥＦＡ.　ＥＮＤＳ.

機能　変数の値に従って、処理の１つを実行します。

書式　ＳＷＩＴＣＨ　変数

ＣＡＳＥ ｛式／文字列｝

実行文

［ＣＡＳＥ ｛式／文字列｝

実行文］

［ＤＥＦＡＵＬＴ

実行文］

ＥＮＤＳＷＩＴＣＨ

参照　ＯＮ～ＧＯＴＯ

説明
- 変数の値をＣＡＳＥの式（文字列）と比較し、一致した場合にそのＣＡＳＥ以降の処理を実行します。１つの処理の範囲は次のＣＡＳＥ、ＤＥＦＡＵＬＴ、もしくはＥＮＤＳＷＩＴＣＨまでです。１つの処理を実行すると、ＥＮＤＳＷＩＴＣＨステートメントに移ります。また、変数の値がどのＣＡＳＥとも一致しなかったときは、ＤＥＦＡＵＬＴに移ります。ＤＥＦＡＵＬＴを省略している場合はＥＮＤＳＷＩＴＣＨに移ります。
- ＳＷＩＴＣＨとＥＮＤＳＷＩＴＣＨは必ず対にして使用します。
- ＣＡＳＥ、ＤＥＦＡＵＬＴ、ＥＮＤＳＷＩＴＣＨは行頭（ラインナンバーのすぐ後ろ）に書く必要があります。なお、ラベルの後ろは行頭にはなりません。行頭に無い場合はＳＷＩＴＣＨで認識されないだけでなく、実行した場合エラーになります。
- ＣＡＳＥに同じ値を指定してもエラーにはなりませんが、実行されるのはＳＷＩＴＣＨに近いほうのＣＡＳＥ処理だけです。
- ＤＥＦＡＵＬＴはＣＡＳＥの処理をすべて書き終わってから一番最後に書かなければなりません。（ＤＥＦＡＵＬＴ処理の後にＣＡＳＥを書くことはできません。）
- ＣＡＳＥまたはＤＥＦＡＵＬＴの実行文中に、ＳＷＩＴＣＨ文は使用できません。
- ＳＷＩＴＣＨ～ＥＮＤＳＷＩＴＣＨの外から内部に飛び込むことはできません。飛び込ませるとエラーになります。

（注）・ＳＷＩＴＣＨ～ＥＮＤＳＷＩＴＣＨから外に飛び出した場合でも、ＳＷＩＴＣＨ処理が終了したことにはなりません。プログラムによっては次のＳＷＩＴＣＨでエラーが発生します。
・ＳＷＩＴＣＨ文では深みをもたせることはできません。なお、スタックは６バイト以上残っていないと使用できません。（379ページのスタック欄参照）
・変数には文字変数や数値変数を使うことができますが、式は使用できません。

〈例〉
```
 10 INPUT"ﾐｾﾉﾅﾏｴ";A$
 20 SWITCH A$
 30  CASE"ABC"
 40   PRINT A$;"TEL:012-3456"
 50  CASE"XYZ"
 60   PRINT A$;"TEL:024-6802"
 70  DEFAULT
 80   PRINT A$;"ﾊ ﾐﾄｳﾛｸ"
 90 ENDSWITCH
100 END
```
このプログラムは、店の電話番号のクイックリファレンスです。中のデータと入力された店名を照合して、合えば店名と電話番号を表示し、合わなければ"～　ﾊ　ﾐﾄｳﾛｸ"と表示して終わります。

基本命令

ＴＲＯＦＦ……トレースオフ
省略形……ＴＲＯＦ.

プログラム　マニュアル

機能　トレースモードを解除します。
書式　ＴＲＯＦＦ
参照　ＴＲＯＮ
説明
● ＴＲＯＮ命令で設定されたトレースモードを解除します。（ＴＲＯＮ参照）

基本命令

ＴＲＯＮ………トレースオン
省略形……ＴＲ.

プログラム　マニュアル

機能　トレースモードを設定します。
書式　ＴＲＯＮ
参照　ＴＲＯＦＦ
説明
● トレースモードでは、プログラムが１行実行されると、その実行された行番号を表示して停止します。
次の行を実行するときは▼を押します。▼でプログラムを１行ずつ実行させていくことができますので、プログラムがどのように実行されていくかをたどることができます。（156ページのデバック参照）
これにより、プログラムが正しく実行されるかどうかをチェックできます。
● トレースモードはＴＲＯＦＦ命令の実行、SHIFT＋CAの操作、電源オフ・オンなどで解除されます。

基本命令

ＵＳＩＮＧ……ユージング
省略形……Ｕ.

プログラム　マニュアル

機能　数値や文字などを表示または印字するときのフォーマットを指定します。
書式　（１）ＵＳＩＮＧ"フォーマット"
　　　（２）ＵＳＩＮＧ
参照　ＰＲＩＮＴ、ＬＰＲＩＮＴ
説明
● ＰＲＩＮＴ命令による表示の形、ＬＰＲＩＮＴ命令による印字の形を指定する命令です。また、マニュアル計算の結果が数値の場合には、このＵＳＩＮＧのフォーマットに従います。
● 表示フォーマットは、書式（１）ではＵＳＩＮＧ命令に続く" "内に、次の記号を用いて指定します。指定の解除は書式（２）で行います。

記号	内容
＃	数値の桁数を指定 整数部桁数：２～11行（符号を含む） 指定桁より数値のほうが少ないときはその分だけスペース表示になり、多いときはエラーになります。12桁以上指定した場合はすべて11桁の指定とみなされます。 小数部桁数：０～12桁（指数方式による表示のときは０～９桁） 指定桁より数値のほうが少ないときはその分だけ０表示になり、多いときはその分だけ切り捨てられます。
．	小数点の表示を指定（整数部と小数部の区切りを指定）
，	数値の３桁区切りの指定 数値の整数部を３桁ごとにコンマ（，）で区切って表示するときは、整数部の＃の途中または最後にコンマを記述します。
∧	数値の指数方式による表示を指定 整数部の指定桁にかかわりなく、仮数部の整数桁は２桁（符号を含む）になります。
＆	文字列の桁数を指定 指定桁より文字数が少ない場合はその分スペース表示になり、文字数が多い場合は指定された桁数分だけ表示します。

〈例〉
① ＵＳＩＮＧ　"＃＃＃"　符号と整数２桁を表示
② ＵＳＩＮＧ　"＃＃＃."　符号と整数２桁と小数点を表示
③ ＵＳＩＮＧ　"＃＃＃.＃＃"　符号と整数２桁と小数点と小数点以下２桁を表示
④ ＵＳＩＮＧ　"＃＃＃，＃＃＃."　符号、３桁区切りマーク（，）、整数５桁と小数点を表示
　３桁区切りマークも１桁と数えます。たとえば、－1,234,567. を表示させるときはＵＳＩＮＧ"＃＃＃＃＃＃，＃＃＃."のように＃と，をあわせて最低10個指定する必要があります。
⑤ ＵＳＩＮＧ　"＃＃.＃＃^"　小数点以下２桁までの指数方式で表示
　［このとき、仮数部の整数は符号と整数１桁、指数部は符号を含めて４桁（Ｅ　００）が自動的に取られます。］
⑥ ＵＳＩＮＧ　"＆＆＆＆＆＆"　文字を６桁表示
⑦ ＵＳＩＮＧ　"＃＃＃＆＆＆＆"　数値と文字を同時に指定
⑧ ＵＳＩＮＧ　フォーマット指定を解除

（注）コンマ（，）とへは混在しては使用できません。

● ＵＳＩＮＧ命令はＰＲＩＮＴ文の中でも使用することができます。

〈例〉
```
10　B＝−10．8：C＝10．7703
20　PRINT　USING　"&&&###"；"B＝"；B，
    "C＝"；USING　"###．###"；C
```

● ＵＳＩＮＧ命令で表示フォーマットの指定を行うと、以降に実行されるＰＲＩＮＴ命令、ＬＰＲＩＮＴ命令には、すべてそのフォーマット指定が有効になります。
したがって、フォーマット指定が不要なときは書式（２）の形で指定を解除してください。
フォーマット指定はＲＵＮ命令の実行や、SHIFT ＋ CA の操作などでも解除されます。

関数

ＶＡＬ…………バリュー

省略形……Ｖ．

プログラム　マニュアル

機能　文字列を数値に変換します。

書式　ＶＡＬ　文字列

参照　ＳＴＲ＄

説明
● 数字（０～９）、符号（＋、－）、指数部を示す記号（Ｅ）で構成されている文字列、および先頭に＆Ｈがついた16進数を表す文字列を数値（10進数）に変換します。
● １つの文字列の中に数値に変換できない文字や記号が含まれている場合、それから右の文字列は無視されます。

〈例〉
```
A＝VAL"−120"      Aに−120が代入されます。
B＝VAL"3．2＊4＝"  Bに3.2が代入されます。
C＝VAL"&HFF"      Cに255が代入されます。
```

関数

ＶＤＥＧ………バリュー・ディグリー

省略形……ＶＤ．

プログラム　マニュアル

機能　60進数（度・分・秒）の文字列を10進数（度）に変換します。

書式　ＶＤＥＧ　文字列

参照　ＤＭＳ＄

説明
● 60進数の文字列を10進数に変換します。

〈例〉
```
10　AA$＝"1°30′36″"
20　B＝VDEG　AA$
30　PRINT　B
```
```
RUN
                1．51
>
```

● 文字列に度（°），分（′），秒（″）の記号が含まれていないときは、整数部を度（°）、小数点以下１～２桁目を分（′），小数点以下３桁目以降は秒（″）とみなします。

〈例〉
```
10　B＝VDEG"1．3036"
20　PRINT　B
```
```
RUN
                1．51
>
```

● 文字列の中に数値に変換できない文字や記号が含まれている場合は、エラー22が発生します。

● “ＶＤＥＧ　ＤＭＳ＄　Ａ”と“ＤＥＧ　ＤＭＳ　Ａ”の結果が異なる場合があります。
これは、ＤＭＳ＄とＤＭＳの有効桁数が違うためです。
ＤＭＳ＄の有効桁数は10桁、ＤＭＳの有効桁数は12桁です。

〈例〉
```
10　A＝1．23456789
20　PRINT　DEG
    DMS　A
30　PRINT　VDEG
    DMS$　A
```
```
RUN
           1．23456789
          1．234566667
>
```

● 度・分・秒の記号のキャラクターコード（くわしくは384ページ参照）
度（°）…223（Ｈ＆ＤＦ）、分（′）…39（Ｈ＆２７）、秒（″）…248（Ｈ＆Ｆ８）

基本命令

ＷＡＩＴ………ウエイト

省略形……Ｗ．

プログラム　マニュアル

機能　ＰＲＩＮＴ命令によるプログラムの停止時間を指定します。

書式　（１）ＷＡＩＴ　式
（２）ＷＡＩＴ

参照　ＰＲＩＮＴ

説明
● 書式（１）では、式で時間を指定すると、ＰＲＩＮＴ命令を実行するたびに指定時間だけプログラムが停止し、その時間が経過すると自動的に再開します。
● 式の値は０～65535の範囲で指定できます。
なお、式の値１は約１／64秒に相当します。
● 時間を無限にしたいときは、書式（２）の形で時間指定を解除してください。この場合、プログラムを再開するには ↵ を押す必要があります。
● 本機の電源を入れたとき、またはＲＵＮ命令を実行したときはＷＡＩＴ　０（停止時間０）に設定されます。
（注）一般のパソコンではＷＡＩＴ指定はできません。パソコンでは通常次のようにＦＯＲ～ＮＥＸＴを用いて、時間の調整を行います。

〈例〉
```
50　FOR　J＝1　TO　500：NEXT　J
```

基本命令

ＷＨＩＬＥ～ＷＥＮＤ…ホワイル～ホワイルエンド

省略形……ＷＨ．　ＷＥ．

プログラム

機能　ＷＨＩＬＥとＷＥＮＤの間に書かれた命令を指定された条件が満たされている間、繰り返し実行します。

書式　ＷＨＩＬＥ　条件式
　実行文１
ＷＥＮＤ
　実行文２

参照　ＦＯＲ～ＮＥＸＴ、ＲＥＰＥＡＴ～ＵＮＴＩＬ

説明
● ＷＨＩＬＥ文において条件式の判断をし、条件が成立しなかった場合は実行文２に移ります。（繰り返しループの終了）。条件が成立している間は実行文１を繰り返し実行します。（この繰

り返し部分をＷＨＩＬＥ～ＷＥＮＤループと呼びます。）

- ＷＨＩＬＥとＷＥＮＤは必ず対にして使用します。
- ＷＨＩＬＥの条件式が最初から成立していない場合は、１回も実行せずにループを抜けます。つまり、実行文１を１回も実行せずに実行文２に移ります。条件式がずっと成立している場合は永遠にループ内の実行を繰り返すプログラムになります。
- ＷＨＩＬＥ～ＷＥＮＤループの中に、別のＷＨＩＬＥ～ＷＥＮＤループを入れることができます。ただし、中に入るループは外のループ内に完全に入っていなければなりません。したがって、ループが交差するような使いかたはできません。（交差している場合はエラーになります。）この条件でループを最大18段まで重ねて使う（深みをもたせる）ことができます。（379ページのスタック参照）
- ＷＨＩＬＥ～ＷＥＮＤループの外からループ内に飛び込むことはできません。飛び込ませるとエラーになります。

（注）・ＷＨＩＬＥ～ＷＥＮＤループから外に飛び出した場合でも、そのループは終了したことにはなりません。プログラムによっては（ＷＨＩＬＥ文を何回か実行するようなプログラムの場合）ＷＨＩＬＥ～ＷＥＮＤの深みエラーが発生することがあります。
・ＷＨＩＬＥ～ＷＥＮＤループ内ではＣＬＥＡＲ、ＥＲＡＳＥ、ＤＩＭ命令は使用できません。

〈例〉
```
 10 PRINT"3 カラ 100 マデ ノ ソスウヲ モトメマス"
 20 DIM A(30):J=0:A=3:A(0)=2
 30 WHILE A<=100
 40   FOR I=0 TO J
 50     IF A-INT(A/A(I))*A(I)=0 THEN
        I=J:NEXT I:GOTO 90
 60   NEXT I
 70   PRINT A
 80   J=J+1:A(J)=A
 90   A=A+1
100 WEND
110 END
```
このプログラムは、３以上の数を２もしくは算出した素数で割り切れるかどうかで素数の判断をしています。Ａ＜＝１００（条件式が成立）の場合は素数を求める計算式を実行し、Ａが１００を超えるまでＷＨＩＬＥ～ＷＥＮＤ間を繰り返し実行します。

基本命令

ＸＯＲ………エクスクルーシブオア

省略形……Ｘ．

［プログラム］［マニュアル］

［機能］式と式との排他的論理和を計算します。

［書式］式　ＸＯＲ　式
条件式　ＸＯＲ　条件式

［参照］ＡＮＤ、ＯＲ、ＮＯＴ、ＩＦ

［説明］
- ２進数において、排他的論理和は次のような値を取ります。

```
1 XOR 1 = 0
1 XOR 0 = 1
0 XOR 1 = 1
0 XOR 0 = 0
```
なお、10進数の排他的論理和を求めた場合でも、計算機内では10進数を２進数に変換したうえで、各桁の排他的論理和を求め、その結果を10進数に戻します。たとえば、41と27の排他的論理和は次のように計算されます。

```
41 XOR 27 = 50
      101001……41
XOR
      011011……27
    ------------
      110010……50
```
41と27をそれぞれ２進数に変換し、各桁のＸＯＲを取ります。そして、その結果を10進数に変換すれば50になります。

- 式の値は－32768～32767の整数部が有効になります。
- ２つ以上の条件のうち奇数個を満足するような条件を１つの式で表します。

〈例〉　５０　ＩＦ　Ａ＞５　ＸＯＲ　Ｂ＞＝４　ＴＨＥＮ…
Ａ＞５またはＢ＞＝４のいずれか一方のみが満足するとき、ＴＨＥＮに続く命令を実行します。

本機には、制御実習のために、周辺機器接続端子（11ピン）を制御できるミニＩ／Ｏ、および８ビット制御に関する命令システムバス端子（40ピン）に対する入出力命令もあります。
これらの命令は本機で制御実習を行うとき以外には使用しないでください。詳しくは、先生の指導に従ってください。

ミニＩ／Ｏに関する命令

ＩＮＰ…………イン・ポート

省略形……なし

［プログラム］［マニュアル］

［機能］入力ポート（Xin，Din，ACK）からの入力関数です。

［書式］ＩＮＰ

［参照］ＯＵＴ

［説明］
- ポート制御命令で、ＩＮＰ命令は現在の入力ポートの状態を読み取って、０～７の値で返します。入力ポートは Xin，Din，ACK の３つからなり、Xin＝４、Din＝２、ACK＝１の重みを持ちます。すなわち、各入力ポートの状態を２進数の各桁と見なしたときの値を、10進数に変換して０～７の値を返します。なお、信号レベルは Hi＝１、Lo＝０とします。

〈例１〉　Xin＝Lo，Din＝Hi，ACK＝Hi のとき
０×４＋１×２＋１×１＝３　で
ＩＮＰの値は３となります。

〈例２〉　５０　ＩＦ　ＩＮＰ　ＡＮＤ　２　ＴＨＥＮ　５０
Din の値が Lo レベルであれば次の行に実行が移り、Hi レベルであれば５０行を繰り返し実行します。

（注）・ＯＰＥＮ"ＣＯＭ１："またはＯＰＥＮ"ＰＩＯ："が実行されているとエラーになります。

ミニＩ／Ｏに関する命令

ＯＵＴ…………アウト・ポート　［プログラム］［マニュアル］

省略形……なし

［機能］ 出力ポート（Busy，Dout，Xout）への出力命令です。

［書式］ ＯＵＴ　式

［参照］ ＩＮＰ

［説明］
● ポート制御命令で、ＯＵＴ命令は10進数（０～７）で指定した値を出力ポートBusy、Dout、Xoutに出力します。指定する値は、出力ポートをそれぞれBusy＝４、Dout＝２、Xout＝１の重みで加算した値です。すなわち、各出力ポートの状態を２進数の各桁と見なしたときの値を10進数で指定します。なお、信号レベルはHi＝１、Lo＝０とします。

〈例〉　ＯＵＴ６
　６＝１×４＋１×２＋０×１　で
　Busy＝Hi，Dout＝Hi，Xout＝Lo　となります。

（注）・ＯＰＥＮ″ＣＯＭ１：″またはＯＰＥＮ″ＰＩＯ：″が実行されているとエラーになります。
・ＯＵＴ命令で出力した内容は、次のＯＵＴ命令が実行されるまで保持されます。ただし、次の操作を行った場合は出力ポートの状態が変化します。

⑴ 次の命令を実行したとき

ＬＰＲＩＮＴ	ＰＲＩＮＴ＃	ＥＮＤ
ＬＬＩＳＴ	ＩＮＰＵＴ＃	ＣＬＯＳＥ
［SHIFT］＋［P↔NP］	ＲＵＮ	ＢＳＡＶＥ
ＢＬＯＡＤ		

⑵ その他、次の操作を行ったとき
・電源を入れた直後
・ＲＵＮ実行直後
・マニュアル計算実行
　ただし、あらかじめＯＰＥＮ″ＬＰＲＴ：″［↵］と操作しておけば出力ポートの状態は保持されます。
・モードを切り替えた場合

ミニＩ／Ｏに関する命令

ＯＰＥＮ…………オープン　［プログラム］［マニュアル］

省略形……ＯＰ.

［機能］ 出力デバイスを指定します。

［書式］ ＯＰＥＮ″ＬＰＲＴ：″

［参照］ ＣＬＯＳＥ、ＬＰＲＩＮＴ、ＬＬＩＳＴ

［説明］
● ＯＰＥＮ″ＬＰＲＴ：″実行後のＬＰＲＩＮＴ、ＬＬＩＳＴ命令は、パラレルポートに対しての出力命令になります。

ミニＩ／Ｏに関する命令

ＣＬＯＳＥ……クローズ　［プログラム］［マニュアル］

省略形……ＣＬＯＳ.

［機能］ デバイス指定を解除します。

［書式］ ＣＬＯＳＥ

［参照］ ＯＰＥＮ、ＬＰＲＩＮＴ、ＬＬＩＳＴ

［説明］
● ＯＰＥＮ状態を解除し、それ以後のＬＰＲＩＮＴ、ＬＬＩＳＴ命令はＣＥ-126Ｐ（プリンタ）に対しての命令になります。

ミニＩ／Ｏに関する命令

ＬＬＩＳＴ……ラインリスト　［マニュアル］

省略形……ＬＬ.

［機能］ プログラム内容をパラレルポートより送出します。

［書式］
（１）ＬＬＩＳＴ
（２）ＬＬＩＳＴ ｛行番号／ラベル｝
（３）ＬＬＩＳＴ　開始行－終了行

［参照］ ＯＰＥＮ、ＣＬＯＳＥ

［説明］
● データ出力命令です。ＯＰＥＮ″ＬＰＲＴ：″により、パラレルポートが指定されているときに、プログラムをアスキーコードで送出します。
● ＣＬＯＳＥされているときは、プリンタ（ＣＥ-126Ｐ）でプログラムを印字する命令になります。（341ページ参照）

ミニＩ／Ｏに関する命令

ＬＰＲＩＮＴ…ラインプリント　［プログラム］［マニュアル］

省略形……ＬＰ.

［機能］ 指定した内容をパラレルポートから送出します。

［書式］ ＬＰＲＩＮＴ［［｛式／文字列｝］［｛，／；｝｛式／文字列｝……］［；］］

［参照］ ＯＰＥＮ、ＣＬＯＳＥ

［説明］
● ＯＰＥＮ″ＬＰＲＴ：″によりパラレルポートが指定されているときに、指定した内容をパラレルポートからアスキーコードで送出します。
● ＣＬＯＳＥ（クローズ）されているときは、プリンタ（ＣＥ-126Ｐ）の印字命令になります。

8ビット制御に関する命令

ＰＩＯＳＥＴ…パラレルセット

省略形……ＰＩ.

プログラム　マニュアル

機能　ミニＩ／Ｏで８ビット制御を行うために各信号の入出力モードを設定します。

書式　ＰＩＯＳＥＴ　式

参照　ＰＩＯＧＥＴ、ＰＩＯＰＵＴ

説明
- 各信号を入力モードにするか、出力モードにするかの設定を行います。
- 式は８ビットに変換され、対応するビットが“１”のときは入力モードに設定され、“０”のときは出力モードに設定されます。

ビット７	ＥＸ２
ビット６	ＥＸ１
ビット５	ＡＣＫ
ビット４	Din
ビット３	Xout
ビット２	Xin
ビット１	Dout
ビット０	Busy

〈例〉　ＰＩＯＳＥＴ　＆ＨＦ０とすると
Din、ＡＣＫ、ＥＸ１、ＥＸ２の端子が入力モードに指定されます。
他は出力モードになります。

- 式は０～255までの値で、この範囲外の値を指定するとエラーになります。
０のときは全ての端子が出力モードになり、255のときは入力モードになります。

8ビット制御に関する命令

ＰＩＯＧＥＴ…パラレルインプット

省略形……ＰＩＯＧ.

プログラム　マニュアル

機能　入力ポートからの８ビット入力関数です。

書式　ＰＩＯＧＥＴ

参照　ＰＩＯＳＥＴ、ＰＩＯＰＵＴ

説明
- ＯＰＥＮ"ＰＩＯ："により８ビット制御が指定されているときに、ＰＩＯＳＥＴ命令で入力モードに設定された端子からのデータを読み取ります。
値は、０～255になります。
- ＰＩＯＳＥＴ命令で出力モードに設定されている端子は、常に“０”になります。

8ビット制御に関する命令

ＰＩＯＰＵＴ…パラレルアウトプット

省略形……ＰＩＯＰ.

プログラム　マニュアル

機能　出力ポートへの出力命令です。

書式　ＰＩＯＰＵＴ　式

参照　ＰＩＯＳＥＴ、ＰＩＯＧＥＴ

説明
- ＯＰＥＮ"ＰＩＯ："により８ビット制御が指定されているときに、ＰＩＯＳＥＴ命令で出力モードに設定された端子からデータを出力します。
１回の命令実行で０～255の範囲内の値を１回だけ返します。
- ＰＩＯＳＥＴで指定した出力端子のうち、式で指定したビット分だけマスクして出力します。

〈例〉　ＰＩＯＳＥＴで15として指定しているとき、ＰＩＯＰＵＴの式が15（00001111）のときは“０”しか出力されず、92（01011100）のときは80（01010000）が出力されます。

- ＰＩＯＳＥＴ命令で入力モードに設定されている端子は、無視されます。

（注）・８ビット制御を使用する場合は、周辺機器接続端子（11ピン）がショートしないようにしてください。（１ｋΩ以下でショートすると故障の原因となります。）

システムバス入出力命令

ＩＮＰ…………イン・ポート

省略形……なし

プログラム　マニュアル

機能　指定したポートから送られてくるデータを読み込みます。

書式　ＩＮＰ　ポートアドレス

説明
- “ポートアドレス”で指定したポートから送られてきたデータを読み込みます。
- “ポートアドレス”は０～６５５３５または＆Ｈ０～＆ＨＦＦＦＦの範囲で指定します。変数や式での指定はできません。
〈例〉　Ａ＝ＩＮＰ　＆Ｈ２０
ポート２０Ｈから送られてきたデータを読み込み、変数Ａに代入します。
- “ポートアドレス”を指定しないときはミニＩ／Ｏに対する命令になります。（366ページ参照）

システムバス入出力命令

ＯＵＴ…………アウト・ポート

省略形……なし

プログラム　マニュアル

機能　指定したポートへデータを送出します。

書式　ＯＵＴ　ポートアドレス，データ

説明
- “ポートアドレス”で指定したポートへ、“データ”で指定した値を出力します。
- “ポートアドレス”は０～６５５３５または＆Ｈ０～＆ＨＦＦＦＦの範囲で指定します。
- “データ”は０～２５５の範囲で指定します。
〈例〉　ＯＵＴ　３２，１２１　　ポート３２（２０Ｈ）へ１２１（７９Ｈ）を出力します。
- “ポートアドレス”を指定しないときはミニＩ／Ｏに対する命令になります。（367ページ参照）

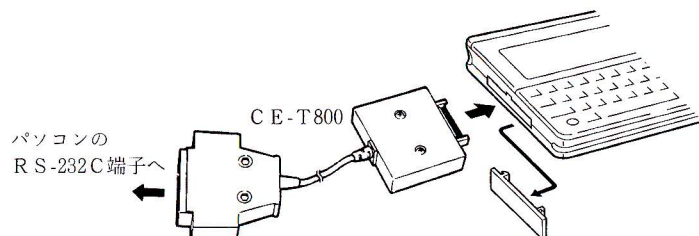
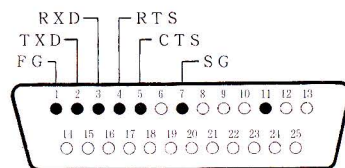
（注）・“ポートアドレス”で指定した値（アドレス）は、アドレスバス（A_0～A_{15}端子）へ出力されます。
“データ”で指定した値はデータバス（D_0～D_7端子）へ出力されます。
・“ポートアドレス”は２０Ｈ～３ＦＨを使用してください。他のアドレスはシステム（本体）で使用しています。

付　　録

1．パソコン通信ケーブルＣＥ－Ｔ８００について

ＣＥ-Ｔ８００は、本機とパソコンなどとの通信（シリアル方式）を行うためのケーブルです。
ＣＥ-Ｔ８００をお持ちの場合、本機とパソコンとの間でＴＥＸＴモードのＳＩＯ機能を使ったプログラムやデータの入出力、機械語モニタを使った機械語の入出力などができます。

シリアル方式によるパソコンとの通信にはパソコンのＲＳ-232Ｃ端子を使用します。本機とＲＳ-232Ｃでは信号の電圧レベルが異なるため、ＲＳ-232Ｃの信号を直接本機に加えると本機が壊れます。
したがって、この信号レベルを合わせるための回路がＣＥ-Ｔ800に入っています。

（注）ＣＥ-Ｔ800の端子に指などで触れないでください。静電気などにより内部回路が壊れることがあります。

【ＣＥ-Ｔ800のコネクタ信号〈ＤＢ-25(Ｗ)〉】

〈図１〉

信号名

ＲＳ-232Ｃ信号			ＣＥ-Ｔ800(〈図１〉)から見た信号の方向	機　能　概　要
ピン番号	信号名	記号		
1	フレーム接地	ＦＧ	——	
2	送信データ	ＴＸＤ(ＳＤ)	入力　パソコン→本機	本機に送信されるデータ信号
3	受信データ	ＲＸＤ(ＲＤ)	出力(注)　本機→パソコン	本機から送信するデータ信号
4	送信要求	ＲＴＳ(ＲＳ)	入力　パソコン→本機	ハイレベル……パソコンからのデータ送信可 ローレベル……パソコンからのデータ送信停止
5	送信可	ＣＴＳ(ＣＳ)	出力(注)　本機→パソコン	受信可のとき……ハイレベル 受信不可のとき……ローレベル
7	信号用接地	ＳＧ	——	入出力装置間の基準電位を合わせます。
11	——	ＮＣ	——	本機では使用していません。

（注）・ＣＥ-Ｔ800は、パソコンのＲＳ-232Ｃ端子につないで使用できるように、パソコンからの出力信号（２番、４番）は本機の入力信号に、パソコンの入力信号（３番、５番）へは本機の出力信号が接続されています。
・これらの出力信号は、下記以外の状態では不定です。
①ＢＡＳＩＣモードおよびＣ言語モードでＳＩＯがオープンしているとき。
②機械語モニタモードでＲ命令およびＷ命令を実行するとき。
③ＴＥＸＴモードでＳＩＯの送受信をするとき。

2．電池の交換について

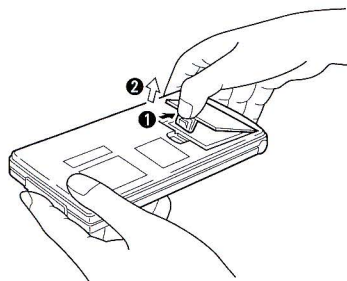
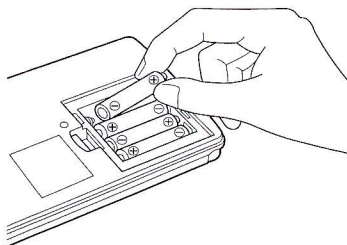
電池が消耗して、規定電圧以下になると画面左下に BATT シンボルが点灯します。
このシンボルが点灯したときは、OFF で電源を切り、再び ON で電源を入れてください。それでもこのシンボルが消えないときは、速やかに次の手順で新しい電池と交換してください。このシンボルが点灯した状態で使用し続けると、本機の電源が自動的に切れて何も動作しなくなります。

（注）機械語命令実行中は、電池が消耗しても BATT シンボルが点灯しませんので、ご注意ください。
機械語命令を実行したまま長時間放置しますと、電圧が低下し、正常な動作をしなくなる恐れがあります。
なお、機械語命令実行中に誤動作等が発生した場合は電池の消耗が考えられます。リセットスイッチで実行を止め、BATT シンボルの点灯または電源が切れる場合は速やかに電池を交換してください。

電池を交換する前に

本機内に大切なプログラムやデータが入っている場合、そのまま電池の交換を行うと、その内容が消えてしまいます。**電池を交換する前に、紙に書き写しておいてください。ＣＥ－１２６Ｐ（プリンタ）をお持ちの場合は、印字しておいてください。**

電池の交換

①本体裏面の電池ぶたを図のようにして外します。

②乾電池を新しいものと交換します。（乾電池は４本とも交換してください。）
乾電池の⊕・⊖をまちがえないように、⊖(マイナス）側から入れます。

③電池ぶたをもとどおり取り付けます。

④14ページの「お買いあげ後はじめてご使用になるときの操作」の(2)〜(3)の項目を実行します。

⑤パソコンなどに記録しておいた内容を読み込みます。

■乾電池は次のタイプをお使いください。
単４形乾電池Ｒ03　４本（指定している電池以外は使用しないでください。）

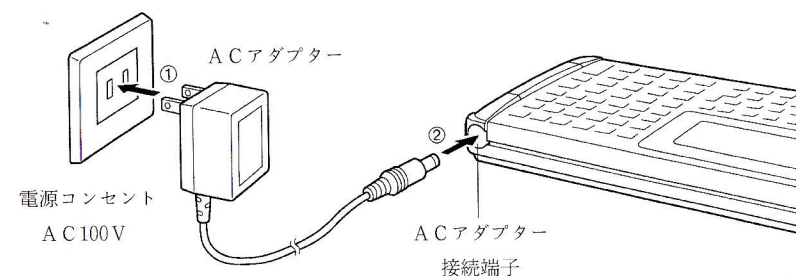
⚠ 電池使用上のご注意

乾電池は誤った使いかたをすると、破れつや発火の原因になることがあります。また、液もれして機器を腐食させたり、手や衣服などを汚す原因になることもあります。以下のことをお守りください。
- 乾電池のプラス（＋）とマイナス（－）の向きを本体の表示どおり正しく入れてください。
- 指定されていない電池を使用しないでください。
- 使えなくなった電池を本体の中に入れたままにしないでください。
- 種類の違うものや、新しいものと古いものを混ぜて使用しないでください。
- もれた液が体についたときは、水でよく洗い流してください。
- 充電や分解、ショートする恐れがあることはしないでください。
また、加熱したり、水や火の中に入れたりしないでください。

使用時間について

付属の電池は工場出荷時に入れていますので、所定の連続使用時間に満たないうちに、寿命が切れることがあります。

ＡＣアダプターの接続のしかた

別売のＡＣアダプターＥＡ-23Ｅを使用すると、家庭用電源“ＡＣ100Ｖ”で動作します。
本機の電源を切ってから、矢印①、②の順に接続してください。

ご注意
- 本体の電池を取り外したまま、ＡＣアダプターだけで動作させないでください。ＡＣアダプターだけで動作させると、誤って接続プラグが外れた場合、せっかく記憶させたデータがすべて消えてしまいます。このため、ＡＣアダプターは本体の電池が消耗しているときは使用できない（動作しない）ことがあります。
- 通信中にＡＣアダプターの抜き差し（コンセントやＡＣアダプター接続端子から）をしないでください。通信が中断されることがあります。

⚠ ＡＣアダプター使用上のご注意

- ＡＣアダプターＥＡ-23Ｅ以外のＡＣアダプターを使用しないでください。故障の原因になります。
- ＡＣアダプターＥＡ-23Ｅを他の機器に使用しないでください。その機器を壊す恐れがあります。

3．主なキーの主な機能

次に主なキーの主な機能を説明します。

キー	機　　能
BREAK ON	●電源を入れるときに押します。 ●プログラム実行中、プログラムを一時停止状態（ＢＲＥＡＫ：ブレーク）にします。 ●ＢＳＡＶＥやＬＯＡＤなどの命令実行中は、その実行を中止します。 ●統計計算では、機能の選択画面に戻すときにも使用します。 ●ＴＥＸＴモード、Ｃ言語機能モードでは、メニュー画面、機能選択画面に戻すときにも使用します。
OFF	●電源を切ります。
BASIC	●ＲＵＮモードとＰＲＯモードを切り替えます。 ●他のモードからＲＵＮモードにします。
SHIFT + ASMBL	●アセンブラ、ＣＡＳＬまたはＰＩＣモードにします。
TEXT	●ＴＥＸＴモードにします。 ●ＴＥＸＴの機能選択画面（メインメニュー画面）にします。
SHIFT + C TEXT	●Ｃ言語機能モードにします。 ●Ｃ言語機能のメニュー画面にします。
SHIFT + コントラスト	●表示の濃度調整画面にします。
SHIFT	●各キーの橙色で書かれている機能を使うとき、このキーを押したまま、それぞれのキーを押します。（2nd F 参照）
小文字 CAPS	●アルファベットの大文字と小文字の入力モードを切り替えます。（画面右側の“ＣＡＰＳ”シンボルの点灯、消灯を行います。） ●カナの入力モードのとき、小さいカナ文字（ァィゥェォャュョッ）を入力したいときに押します。（“小”シンボルの点灯、消灯を行います。）
カナ	●カナ入力モードの設定、解除を行います。（“カナ”シンボルの点灯、消灯を行います。）
TAB	●カーソルを決められた桁数だけ移動させます。 ＲＵＮ、ＰＲＯモードでは７桁ずつ移動します。ＴＥＸＴモードのエディタでは、最初８桁、２回目は６桁、３回目以降は７桁ずつ移動させます。
▶	●カーソルを右に移動させます。 ●プレイバックを行います。 ●プログラムが表示されているときで、カーソルが表示されていない場合はカーソルを呼び出します。 ●マニュアル計算などでのエラーを解除します。
◀	●カーソルを左に移動させます。 ●その他は ▶ と同じ。
ANS	●ラストアンサーを呼び出します。
CONST	●定数計算の定数と計算命令を設定します。（“ＣＯＮＳＴ”シンボル点灯） 2nd F CONST（SHIFT + CONST）を押すと、設定されている定数を表示します。

キー	機　　能
INS	●訂正モードと挿入モードの切り替えを行います。
SHIFT + DEL	●現在のカーソル位置の文字を削除します。
BS	●カーソル位置の１字前の文字を削除します。
2nd F	●各キーの橙色で書かれている機能を使うとき、各キーを押す前にこのキーを押します。（SHIFT 参照）
CLS	●入力中の内容や表示をクリア（消去）するときに押します。 ●エラーを解除します。
SHIFT + CA	●計算機の状態を解除します。（クリアオール） ・プログラムの実行が一時停止状態にあるとき、実行を中止させます。 ・表示内容などを消去します。 ・表示フォーマット指定（ＵＳＩＮＧ指定）を解除します。 ・トレースモードを解除（ＴＲＯＦＦ状態に）します。 ・エラーを解除します。 その他
↵	●プログラムの行の終了を指定します。 ●プログラムを計算機に書き込みます。 ●マニュアル計算の実行、あるいはＢＡＳＩＣ命令などのマニュアル操作による実行を行います。 ●ＰＲＩＮＴ命令やＩＮＰＵＴ命令で一時停止しているプログラムの再スタートなど、プログラムの再開に使用します。
SHIFT + P↔NP	●プリンタが接続され、電源が入っているとき、プリントモードの設定、解除を行います。（“ＰＲＩＮＴ”シンボルの点灯、消灯を行います。）

▼▲の動きは、モードの指定および計算機の状態によって変わります。ＢＡＳＩＣモード（ＲＵＮ、ＰＲＯ）では次のようになります。

モード	状　態	▼	▲
ＲＵＮ	プログラム実行中	無効	無効
	プログラムの一時停止中、ＷＡＩＴが無限のＰＲＩＮＴ命令実行中やＩＮＰＵＴ命令実行中のブレーク状態	次の行を実行（１行ずつ実行して停止します）	押している間、実行している行あるいは実行した行を表示
	プログラム実行時のエラー状態	無効	押している間、エラーが発生した行を表示
	トレースモードオン状態	トレースを実行	押している間、実行している行あるいは実行した行を表示
ＰＲＯ	（ＲＵＮモードからＰＲＯモードに切り替え、プログラムが表示されていないとき）		
	プログラムの一時停止中	停止している行を表示	同　左
	エラー発生後	エラーが発生した行を表示	同　左
	その他	先頭行を表示	最終行を表示
ＰＲＯ	（プログラム行が表示されているとき）		
		次のプログラム行を表示	１行前の行を表示

4．計算範囲

加減乗除算

被演算数、演算数、結果が$\pm 1\times 10^{-99}\sim\pm 9.999999999\times 10^{99}$および0

関数計算

関数	計算範囲	備考
SIN x COS x TAN x	DEG　：$\lvert x\rvert < 1\times 10^{10}$ RAD　：$\lvert x\rvert < \frac{\pi}{180}\times 10^{10}$ GRAD：$\lvert x\rvert < \frac{10}{9}\times 10^{10}$ ただし tan x において次の場合は除く DEG　：$\lvert x\rvert = 90(2n-1)$ RAD　：$\lvert x\rvert = \frac{\pi}{2}(2n-1)$ GRAD：$\lvert x\rvert = 100(2n-1)$ （n は整数）	
ASN x ACS x	$-1\leqq x\leqq 1$	$\sin^{-1}x$ $\cos^{-1}x$
ATN x	$\lvert x\rvert < 1\times 10^{100}$	$\tan^{-1}x$
HSN x HCS x HTN x	$-227.9559242\leqq x\leqq 230.2585092$	$\sinh x$ $\cosh x$ $\tanh x$
AHS x	$\lvert x\rvert < 1\times 10^{50}$	$\sinh^{-1}x$
AHC x	$1\leqq x < 1\times 10^{50}$	$\cosh^{-1}x$
AHT x	$\lvert x\rvert < 1$	$\tanh^{-1}x$
LN x LOG x	$1\times 10^{-99}\leqq x < 1\times 10^{100}$	$\ln x = \log_e x$
EXP x	$-1\times 10^{100} < x\leqq 230.2585092$	e^x　$e\fallingdotseq 2.718281828$
TEN x	$-1\times 10^{100} < x < 100$	10^x
RCP x SQU x CUB x SQR x CUR x	$\lvert x\rvert < 1\times 10^{100}$　$x\neq 0$ $\lvert x\rvert < 1\times 10^{50}$ $\lvert x\rvert < 2.154434690\times 10^{33}$ $0\leqq x < 1\times 10^{100}$ $\lvert x\rvert < 1\times 10^{100}$	$\frac{1}{x}$ x^2 x^3 $\sqrt{x}$ $\sqrt[3]{x}$
$y\wedge x$ (y^x)	● $y>0$ のとき $-1\times 10^{100} < x\log y < 100$ ● $y=0$ のとき $x>0$ ● $y<0$ のとき x は整数または $\frac{1}{x}$ が奇数 ただし$-1\times 10^{100} < x\log\lvert y\rvert < 100$	$y^x = 10^{x\cdot\log y}$

関数	計算範囲	備考
&H x	$0\leqq x\leqq$ 2540BE3FF　または FDABF41C01$\leqq x\leqq$ FFFFFFFFFF	x は16進数での整数
POL (x,y) $(x,y\to r,\theta)$	$(x^2+y^2) < 1\times 10^{100}$ $\frac{x}{y} < 1\times 10^{100}$	$r=\sqrt{x^2+y^2}$ $\theta=\tan^{-1}\frac{y}{x}$
REC (r,θ) $(r,\theta\to x,y)$	$r < 1\times 10^{100}$ θ の範囲は $\sin x$、$\cos x$ の x と同じ	$x=r\cos\theta$ $y=r\sin\theta$
NPR (n,r)	$\frac{n!}{(n-r)!} < 1\times 10^{100}$　$0\leqq r\leqq n\leqq 9999999999$ r、n は整数	${}_nP_r$
NCR (n,r)	$\frac{n!}{(n-r)!r!} < 1\times 10^{100}$　$0\leqq r\leqq n\leqq 9999999999$ r、n は整数 $n-r<r$ のとき　$n-r\leqq 69$ $n-r\geqq r$ のとき　$r\leqq 69$	${}_nC_r$
FACT x	$0\leqq x\leqq 69$	$n!$
DEG x	$\lvert x\rvert < 1\times 10^4$	DMS→DEG
DMS x	$\lvert x\rvert < 1\times 10^4$	DEG→DMS

統計計算

	計算範囲
データ	$\lvert x\rvert < 1\times 10^{50}$　$1\leqq n < 1\times 10^{100}$　$\lvert y\rvert < 1\times 10^{50}$
統計量	次の計算において、計算結果および途中結果の絶対値が1×10^{100}未満であること。 分母（除数）が0でないこと。$\sqrt{\quad}$ で計算する値が負数でないこと。 Σx　Σx^2　Σy　Σy^2 $\bar{x}=\frac{\Sigma x}{n}$　$\bar{y}=\frac{\Sigma y}{n}$ $sx=\sqrt{\frac{\Sigma x^2-n\bar{x}^2}{n-1}}$　$sy=\sqrt{\frac{\Sigma y^2-n\bar{y}^2}{n-1}}$ $\sigma x=\sqrt{\frac{\Sigma x^2-n\bar{x}^2}{n}}$　$\sigma y=\sqrt{\frac{\Sigma y^2-n\bar{y}^2}{n}}$ $a=\bar{y}-b\bar{x}$　$b=\frac{Sxy}{Sxx}$ $r=\frac{Sxy}{\sqrt{Sxx\cdot Syy}}$　$Sxx=\Sigma x^2-\frac{(\Sigma x)^2}{n}$ $x'=\frac{y-a}{b}$　$Syy=\Sigma y^2-\frac{(\Sigma y)^2}{n}$ $y'=a+bx$　$Sxy=\Sigma xy-\frac{\Sigma x\cdot\Sigma y}{n}$

計算の誤差は原則として、10桁目に±1となります。（指数表示の場合は仮数部表示の最下位桁に±1となります。）

ただし、関数の特異点および変曲点の近くでは誤差が累積されて大きくなります。また、連続計算を行った場合もそれぞれの誤差が累積されて大きくなります。（べき乗（y^x）のように、計算機内で連続計算を行っている場合も同様です。）

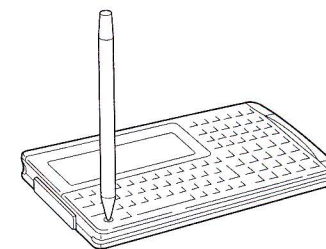
5．仕様

形　　名	ＰＣ-Ｇ850ＶＳ
Ｃ　Ｐ　Ｕ	ＣＭＯＳ　8ビットＣＰＵ（Ｚ80相当品）
メモリ容量	システムエリア……………………………約2.3Ｋバイト データ専用エリア………………………………208バイト プログラム・データエリア…………………30179バイト
スタック	ファンクション用　16段　　データ用　8段　　サブルーチン用　10段 構造化ＢＡＳＩＣ用　　合計90バイト ［ＲＥＰＥＡＴ～ＵＮＴＩＬ：1段で4バイト ＷＨＩＬＥ～ＷＥＮＤ：　1段で5バイト ＳＷＩＴＣＨ～ＣＡＳＥ：　1段で6バイト ＦＯＲ～ＮＥＸＴ：　1段で18バイト （ただし、ＳＷＩＴＣＨ～ＣＡＳＥは1段しか使用できません。）
基本計算機能	基本計算：加減乗除算 関数計算：三角関数、逆三角関数、対数、指数、角度変換、べき乗、開平計算、整数化、絶対値、符号関数、円周率、座標変換、その他
計算桁数	10桁（仮数部）＋2桁（指数部）
計算方法	数式どおり（優先順位判別機能つき）
周辺機器接続端子（11ピン）	ＣＥ-126Ｐ（プリンタ）、ＣＥ-Ｔ800（パソコン通信ケーブル）、ＥＡ-129Ｃ（ポケコン接続ケーブル）
システムバス端子（40ピン）	
表　　示	液晶表示　5×7ドットマトリックス表示（24桁×6行）
使用温度	0℃～40℃
電　　源	6Ｖ⎓（ＤＣ）：単4形乾電池Ｒ03　4本 （ＡＣ100Ｖ 50／60Hz：別売のＡＣアダプターＥＡ-23Ｅ使用）
電池使用時間	実使用状態で連続使用　約90時間（単4形乾電池Ｒ03の場合） ［使用温度25℃で1時間当り演算またはプログラム実行を10分間、表示状態を50分間行った場合］ （注）パソコン通信ケーブルＣＥ-Ｔ800を使用して通信を行っているときの電池使用時間は約70時間になります。 ［使用温度25℃で、通信を2分間、演算またはプログラム実行を8分間、表示状態を50分間行った場合］ ●電池の種類や使用方法などにより多少の変動があります。
消費電力	約0.2W
外形寸法	幅196㎜×奥行95㎜×厚さ20㎜
質　　量	約260ｇ（乾電池含む。ハードカバーは除く。）
付属品	ハードカバー、単4形乾電池　4本、ネームラベル（本体裏面に貼りつけ済み）、取扱説明書※、お客様ご相談窓口のご案内 ※当商品は日本国内向けであり、日本語以外の説明書はございません。 This model is designed exclusively for Japan, with manuals in Japanese only.

異常が発生した場合の処理

BREAK ［ON］を含めたすべてのキーの機能が働かない、あるいは正しく動作しないなどの異常が発生した場合は、本機の電源を入れたままで周辺機器の接続や取り外しを行ったか、またはプログラムのミス、あるいは強度の外来ノイズなどによる異常発生などの原因が考えられます。
または、電池交換を行った後、リセットスイッチを押さなかった場合が考えられます。
このような場合は、次の方法でリセットしてください。

リセットのしかた

①［ON］を押して電源を入れた後、ボールペンなどで、本体左端のリセット（ＲＥＳＥＴ）スイッチを押してください。
芯先の出たシャープペンシルや先の折れやすいもの、また、針など先のとがったものは使用しないでください。

②リセットスイッチを離すと次の画面になります。違う画面になったときはもう一度リセットスイッチを押してください。

MEMORY CLEAR O. K. ? (Y/N)

（メモリー内容を消去しますか？）

③プログラムやデータを保持したいときは、
［N］を押してください。ＲＵＮモードの最初の画面になります。

● この後、プログラム実行などで再び異常が発生する場合は、次のプログラムやデータなどをすべて消去する方法を行ってください。

プログラムやデータなどをすべて消去するときは、
上記の表示中に［Y］を押してください。記憶内容をすべて消去して、次の画面（点滅）になります。
（初期設定し、記憶内容をすべて消去したことを示しています。）

この画面で、［↵］を押せばＲＵＮモードの最初の画面になります。

システムバス端子信号表

表面		裏面	
信号名	端子番号	端子番号	信号名
Vcc	1	2	Vcc
$\overline{M1}$	3	4	$\overline{MREQ}$
$\overline{IORQ}$	5	6	IORESET
$\overline{WAIT}$	7	8	$\overline{INT1}$
$\overline{WR}$	9	10	$\overline{RD}$
BNK1	11	12	BNK0
$\overline{CEROM2}$	13	14	CERAM2
D7	15	16	D6
D5	17	18	D4
D3	19	20	D2
D1	21	22	D0
A15	23	24	A14
A13	25	26	A12
A11	27	28	A10
A9	29	30	A8
A7	31	32	A6
A5	33	34	A4
A3	35	36	A2
A1	37	38	A0
GND	39	40	GND

システムバス端子カバーを取り外すとき

上にスライドさせる。

システムバス端子カバーを取り付けるとき

みぞにあわせてスライドさせる。

(注) 本機の Vcc 電源電圧は、電池の消耗度合いにより、4～6Vの間で変動します。
本機内部は、C-MOS部品により構成されているため、端子に Vcc +0.5V 以上の信号電圧が加えられると、本機内部が壊れることがあります。

ローマ字→カナ変換表

	A		I		U		E		O	
ア行	A	ア	I	イ	U	ウ	E YE	エ イェ	O	オ
カ行	KA CA QA KYA	カ カ クァ キャ	KI QI KYI	キ クィ キィ	KU CU QU KYU	ク ク ク キュ	KE QE KYE	ケ クェ キェ	KO CO QO KYO	コ コ クォ キョ
サ行	SA SHA SYA	サ シャ シャ	SI SHI SYI	シ シ シィ	SU SHU SYU	ス シュ シュ	SE SHE SYE	セ シェ シェ	SO SHO SYO	ソ ショ ショ
タ行	TA TSA CHA TYA CYA	タ ツァ チャ チャ チャ	TI TSI CHI TYI CYI	チ ツィ チ チィ チィ	TU TSU CHU TYU CYU	ツ ツ チュ チュ チュ	TE TSE CHE TYE CYE	テ ツェ チェ チェ チェ	TO TSO CHO TYO CYO	ト ツォ チョ チョ チョ
ナ行	NA NYA	ナ ニャ	NI NYI	ニ ニィ	NU NYU	ヌ ニュ	NE NYE	ネ ニェ	NO NYO	ノ ニョ
ハ行	HA FA HYA	ハ ファ ヒャ	HI FI HYI	ヒ フィ ヒィ	HU FU HYU	フ フ ヒュ	HE FE HYE	ヘ フェ ヒェ	HO FO HYO	ホ フォ ヒョ
マ行	MA MYA	マ ミャ	MI MYI	ミ ミィ	MU MYU	ム ミュ	ME MYE	メ ミェ	MO MYO	モ ミョ
ヤ行	YA	ヤ	YI	イ	YU	ユ			YO	ヨ
ラ行	RA LA RYA LYA	ラ ラ リャ リャ	RI LI RYI LYI	リ リ リィ リィ	RU LU RYU LYU	ル ル リュ リュ	RE LE RYE LYE	レ レ リェ リェ	RO LO RYO LYO	ロ ロ リョ リョ
ワ行	WA	ワ							WO	ヲ
ン	N	N SHIFT + ’(U)			M					

	A		I		U		E		O	
ア行	VA	ヴァ	VI	ヴィ	VU	ヴ	VE	ヴェ	VO	ヴォ
ガ行	GA	ガ	GI	ギ	GU	グ	GE	ゲ	GO	ゴ
	GYA	ギャ	GYI	ギィ	GYU	ギュ	GYE	ギェ	GYO	ギョ
ザ行	ZA	ザ	ZI	ジ	ZU	ズ	ZE	ゼ	ZO	ゾ
	JA	ジャ	JI	ジ	JU	ジュ	JE	ジェ	JO	ジョ
	JYA	ジャ	JYI	ジィ	JYU	ジュ	JYE	ジェ	JYO	ジョ
	ZYA	ジャ	ZYI	ジィ	ZYU	ジュ	ZYE	ジェ	ZYO	ジョ
ダ行	DA	ダ	DI	ヂ	DU	ヅ	DE	デ	DO	ド
	DHA	デャ	DHI	ディ	DHU	デュ	DHE	デェ	DHO	デョ
	DYA	ヂャ	DYI	ヂィ	DYU	ヂュ	DYE	ヂェ	DYO	ヂョ
バ行	BA	バ	BI	ビ	BU	ブ	BE	ベ	BO	ボ
	BYA	ビャ	BYI	ビィ	BYU	ビュ	BYE	ビェ	BYO	ビョ
パ行	PA	パ	PI	ピ	PU	プ	PE	ペ	PO	ポ
	PYA	ピャ	PYI	ピィ	PYU	ピュ	PYE	ピェ	PYO	ピョ

①「ン」の入力

「ン」は「N」と入力します。

ただし、「N」の次に母音（A、I、U、E、O）および「Y」がくるとき、または後に文字がこないときは「N [SHIFT]＋[U]」と押します。

なお、「N」の代わりに「M」を用いることもできます。

（例）　シンユウ ──→ SIN [SHIFT]＋[U] YUU

　　　　シンニュウ ──→ SINNYUU

　　　　シンブン ──→ SIMBUN [SHIFT]＋[U]

②「ッ」の入力

「ッ」は子音を重ねて入力します。ただし、「N」や「M」を重ねて入力したときは「ン」が入ります。

（例）　キップ ──→ KIPPU

　　　　セット ──→ SETTO

③小さい文字の単独入力

小さい文字（ァ、ィ、ゥ、ェ、ォ、ッ、ャ、ュ、ョ）を入力するときは [CAPS]（小文字）を押してからそれぞれの文字を入力します。

（例）　ティーカップ ──→ TE [CAPS]（小文字） I－KAPPU

キャラクタ・コード表

下位桁 ＼ 上位桁		0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
	16進数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	ヌル		スペース	0	@	P	‘	p		⊥		ー	タ	ミ	＝	×
1	1			!	1	A	Q	a	q	▁	⊤	。	ア	チ	ム	╞	円
2	2			″	2	B	R	b	r	▂	┤	「	イ	ツ	メ	╪	年
3	3			#	3	C	S	c	s	▃	├	」	ウ	テ	モ	╡	月
4	4			$	4	D	T	d	t	▄	‾	、	エ	ト	ヤ	◢	日
5	5			%	5	E	U	e	u	▅	─	・	オ	ナ	ユ	◣	時
6	6			&	6	F	V	f	v	▆	│	ヲ	カ	ニ	ヨ	◥	分
7	7			’	7	G	W	g	w	▇	▕	ァ	キ	ヌ	ラ	◤	秒
8	8			(	8	H	X	h	x	▏	┌	ィ	ク	ネ	リ	♠	″
9	9			)	9	I	Y	i	y	▎	┐	ゥ	ケ	ノ	ル	♥	
10	A			*	:	J	Z	j	z	▍	└	ェ	コ	ハ	レ	◆	
11	B			+	;	K	[	k	{	▌	┘	ォ	サ	ヒ	ロ	♣	
12	C			,	<	L	¥	l	¦	▋	╭	ャ	シ	フ	ワ	●	
13	D			−	=	M	]	m	}	▊	╮	ュ	ス	ヘ	ン	○	
14	E			.	>	N	∧	n	～	▉	╰	ョ	セ	ホ	゛	／	
15	F			／	?	O	_	o		＋	╯	ッ	ソ	マ	゜	＼	

キャラクタコードは次のように表します。

〈例〉　“＊”のコード

　16進数　&H2A

　10進数　42（32＋10）

［補足］

CHR$命令により、本機でキャラクタを表示させる場合

- 表中のコード0（&H00）のキャラクタはヌル（Null：何もない状態）です。したがって何も表示されません。
- キャラクタが記載されていない部分はスペース（空）になります。

CHR$命令により、CE-126Pでキャラクタを印字させる場合

- キャラクタが記載されていない部分はスペース（空）になります。
- 次のコードはスペースになります。
  129(&H81)～159(&H9F)、224(&HE0)～231(&HE7)、236(&HEC)～240(&HF0)、245(&HF5)～248(&HF8)

メモリマップ・I／Oマップ

メモリマップ

アドレス	エリア		
0000H	リザーブエリア		
0100H	機械語エリア		
	プログラムファイルエリア		
	ラムデータファイルエリア		
	テキストエリア		
	ＢＡＳＩＣプログラムエリア（プログラム・データエリア）		
	変数エリア		
	ワークエリア		
	固定変数エリア		
	ワークエリア		
	スタックエリア		
8000H	ROM　BANK　0		
C 000H	ROM　BANK　1	ROM BANK　2	ROM BANK　3
ＦＦＦＦH			

I／Oマップ

アドレス	内容
00H	使用不可
10H	システムポート
20H	空きポート
30H	空きポート
40H	ディスプレイ用
50H	ディスプレイ用
60H	システムポート
70H	システムポート
80H	使用不可
FFH	

エラーコード表

エラーコード	内　容
10	文法的に実行できない場合。
12	ＰＲＯモードでしか使えない命令（ＬＩＳＴ、ＲＥＮＵＭなど）をＲＵＮモードで使おうとした。または、逆にＲＵＮモードでしか使えない命令をＰＲＯモードで使おうとした。 ＯＰＥＮ命令のモード指定がまちがっている。
13	ＣＯＮＴ命令でプログラムの再実行ができない。
14	ＢＡＳＩＣプログラムがないときに、パスワードを設定しようとした。
15	ＢＳＡＶＥ　Ｍ命令で、アドレスの指定が逆。（開始アドレスよりも、終了アドレスの方が小さくなっている。）
20	計算結果が 1×10^{100} 以上になった。
21	除数が０の除算を実行した。
22	不合理な演算を行った。（例　ＬＯＧ－３） 計算範囲外の値の関数計算を行った。（例　ＡＳＮ　２）
30	すでに宣言されている配列変数名を再度宣言している。
31	ＤＩＭ命令で宣言していない配列変数を使用している。
32	配列変数の添字がＤＩＭ命令で宣言した大きさを超えている。
33	指定している数値が規定の範囲から外れている。
40	指定した行番号やラベルが存在しない。
41	行番号（ラインナンバー）が１～65279の範囲外になっている。
43	ＲＥＮＵＭ、ＬＣＯＰＹ命令の指定に不都合がある。 （指定した開始行以降の行番号のつけ替えを行うと、開始行よりも小さい番号の行（行番号のつけ替えをしない行）と混ざってしまうような指定になっている。）
44	ＬＬＩＳＴ、ＤＥＬＥＴＥなどの命令で指定した開始行と終了行の大きさが逆になっている。（終了行よりも開始行が大きくなっている。）
50	ＧＯＳＵＢ、ＦＯＲ、ＲＥＰＥＡＴ、ＷＨＩＬＥもしくはＳＷＩＴＣＨでの段数オーバー。
51	ＲＥＴＵＲＮ文に対するＧＯＳＵＢ文がない。
52	ＮＥＸＴ文に対するＦＯＲ文がない。
53	ＲＥＡＤ文に対するＤＡＴＡ文がない。
54	ファンクションバッファ（16段）または、データバッファ（８段）の段数オーバー。
55	文字記憶エリア（255文字）の容量を超えた。 １行の長さが255バイトを超えた。
60	プログラムおよび変数が大きすぎて、プログラム・データエリアの容量を超えた。
61	ブロック形式のＥＮＤＩＦがないのにブロック形式のＩＦ文もしくはブロック形式のＥＬＳＥ文を実行した。
62	ＵＮＴＩＬに対するＲＥＰＥＡＴがない。
63	ＷＨＩＬＥに対するＷＥＮＤがない。

エラーコード	内　容
64	ＷＥＮＤに対するＷＨＩＬＥがない。
66	ＤＥＦＡＵＬＴを実行中にＣＡＳＥまたはＤＥＦＡＵＬＴを実行しようとした。
68	ＳＷＩＴＣＨ、ＣＡＳＥもしくはＤＥＦＡＵＬＴに対するＥＮＤＳＷＩＴＣＨがない。
69	ＳＷＩＴＣＨが実行されていないのにＣＡＳＥ、ＤＥＦＡＵＬＴまたはＥＮＤＳＷＩＴＣＨを実行した。
70	ＵＳＩＮＧ命令で指定されたフォーマットで表示または印字できない。
71	ＵＳＩＮＧ命令の指定が正しくない。（"　"で囲んで指定した内容が正しくない。）
72	入出力装置に関するエラー。
77	ファイルの容量が足りない。
80	ＳＩＯに対するリードイン（読み込み）エラー。
81	ＳＩＯ、ミニＩ／Ｏなどのタイムアウトエラー（プログラムやデータの入出力で、規定の待ち時間を超えた。）
82	ＢＬＯＡＤ　？命令による内容の照合で、内容の不一致がある。
83	ＩＮＰＵＴ＃命令で指定している変数の型が、読み込もうとしているデータの型と一致していない。
84	プリンタ関連のエラー。
85	ＰＲＩＮＴ＃、ＩＮＰＵＴ＃命令でデータの入出力を行うとき、相手の装置（デバイス）がオープンされていない。
86	１つの装置（デバイス）がオープンしているときに、同じファイル番号でほかの装置をオープンしようとした。または、ファイルがすでにオープンされている。
87	ファイルのデータを最後まで読み込んだ後、さらにデータを読み込もうとした。
90	数値変数に文字、文字変数に数値を代入しようとした。また、ＳＩＮ　Ａ＄　のように数値を扱う関数に文字変数を指定したなど、変数名の不適合。
91	固定変数において、数値が入っている変数を文字変数として使用したり、文字が入っている変数を数値変数として使用した。
92	パスワードが一致していない。
93	パスワードが設定されているときに、マニュアルでＭＯＮ、ＰＥＥＫ、ＰＯＫＥ、ＲＥＮＵＭなどの命令を実行しようとした。
94	指定されたファイルが存在しない。
95	ファイル名の指定が正しくない。
96	ＢＡＳＩＣモードでＴＥＸＴファイルを読み込もうとした。
97	ファイルの数が255を超えた。

Ｃ言語でのコンパイル時と実行時のエラーメッセージについては241ページを参照してください。

故障かな？と思ったら

次のような場合は故障でないことがありますので、修理を依頼される前にもう一度お確かめください。それでも具合の悪いときは、次ページの「アフターサービスについて」をご覧のうえ修理を依頼してください。

こんなとき	ここをお確かめください
電源が入らない	● 新しい電池と交換してください。(☞372ページ) ● リセットしてください。(☞380ページ)
表示が薄いまたは濃い	● 表示濃度を調整してください。(☞17ページ) ● "BATT" が点灯しているときは、電池を交換してください。(☞372ページ)
すべてのキーが働かないまたは正しく働かない	● リセットしてください。(☞380ページ)
プログラムが表示できない	● パスワードを解除してください（ＰＡＳＳ命令でプログラムの表示を禁止しているとき）。(☞348ページ) ● 最初からプログラムを入れ直してください（何かの原因でプログラムが消えているとき）。
プリンタで印字できない	● ＲＵＮモードで、ＰＯＫＥ　＆Ｈ７７７４，０を実行してください（メモリーの一部が、本機の内部制御コードと偶然一致したとき）。

アフターサービスについて

保証について

1. **この製品には取扱説明書の巻末に保証書がついています。**
   保証書は販売店にて所定事項を記入してお渡しいたしますので、内容をよくお読みのうえ大切に保存してください。
2. **保証期間はお買いあげの日から３年間です。**
   保証期間中でも有料になることがありますので、保証書をよくお読みください。
3. **保証期間後の修理は……**
   修理によって機能が維持できる場合は、ご要望により有料修理いたします。

補修用性能部品の保有期間

- 当社はポケットコンピュータの補修用性能部品を製品の製造打切後７年保有しています。
- 補修用性能部品とは、その製品の機能を維持するために必要な部品です。

修理を依頼されるときは

1. 異常があるときは使用をやめて、お買いあげの販売店にこの製品を お持込み のうえ、修理をお申しつけください。**ご自分での修理はしないでください。**
2. **アフターサービスについてわからないことは……**
   お買いあげの販売店、またはもよりのシャープお客様ご相談窓口にお問い合わせください。
3. ポケットコンピュータに接続した周辺機器に起因して、ポケットコンピュータ本体が故障・損傷した場合は、保証書に記載されている無料修理規定が適用されません（有料修理となります）ので、あらかじめご了承ください。

お問い合わせは

この製品についてのご意見、ご質問は、もよりのシャープお客様ご相談窓口へお申しつけください。
付属の「お客様ご相談窓口のご案内」のとおり、全国にお客様ご相談窓口を設けております。

＝メ　モ＝

=メ　モ=

SHARP

ポケットコンピュータ保証書 持込修理
(WARRANTY CARD)

本書は、記載内容の範囲で無料修理をさせていただくことをお約束するものです。保証期間中に故障が発生した場合は、商品と本書をご持参のうえ、お買いあげの販売店にご依頼し本書をご提示ください。お買いあげ年月日、販売店名など記入もれがありますと無効です。記入のない場合は、お買いあげの販売店にお申し出ください。
ご転居・ご贈答品などでお買いあげの販売店に修理をご依頼できない場合は、商品に同梱しております「お客様ご相談窓口のご案内」をご覧のうえ、お客様ご相談窓口にお問い合わせください。
本書は再発行いたしません。たいせつに保管してください。

〈無料修理規定〉

1. 取扱説明書・本体注意ラベルなどの注意書にしたがった正常な使用状態で、保証期間内に故障した場合には、お買いあげの販売店が無料修理いたします。
   なお、故障の内容によりまして、修理にかえ同等製品と交換させていただくことがあります。
2. 保証期間内でも、次の場合には有料修理となります。
   (イ) 本書のご提示がない場合。
   (ロ) 本書にお買いあげ年月日・お客様名・販売店名の記入がない場合、または字句を書き換えられた場合。
   (ハ) 使用上の誤り、または不当な修理や改造による故障・損傷。
   (ニ) お買いあげ後に落とされた場合などによる故障・損傷。
   (ホ) 火災・公害および地震・雷・風水害その他天災地変など、外部に要因がある故障・損傷。
   (ヘ) 消耗部品(乾電池)が損耗し、取り替えが必要な場合。
   (ト) 電池の液漏れ、または、指定規格外の電池の使用による故障・損傷。
   (チ) 持込修理の対象商品を直接メーカーへ送付した場合の送料等はお客様の負担となります。また、出張修理等を行った場合には、出張料はお客様の負担となります。
3. 本書は日本国内においてのみ有効です。
   (THIS WARRANTY CARD IS ONLY VALID FOR SERVICE IN JAPAN.)

★この保証書は本書に明示した期間・条件のもとにおいて無料修理をお約束するものです。したがいまして この保証書によって保証書を発行している者（保証責任者）、および、それ以外の事業者に対するお客様の法律上の権利を制限するものではありませんので、保証期間経過後の修理などにつきましておわかりにならない場合は、お買いあげの販売店またはシャープお客様ご相談窓口にお問い合わせください。

修理メモ

形名	PC-G850VS	
お客様	ふりがな お名前　　　　様 ☎	
	〒 ご住所	
取扱販売店名・住所・電話番号		
保証期間	お買いあげ日 年　月　日より	**本体は3年間** （消耗部品は除く）

シャープ株式会社

〒545-8522　大阪市阿倍野区長池町22-22

お問合せ先：お客様相談センター　フリーダイヤル 0120-303-909

フリーダイヤルが使用できない場合のご利用は　043-351-1822

または　06-6792-1583

■よくある質問などはパソコンから検索できます。

http://www.sharp.co.jp/support/

シャープ　お問い合わせ

使用方法・お買い物相談など

【お客様相談センター】

0120 - 303 - 909

携帯電話・PHSからもご利用いただけます。

■IP電話などからフリーダイヤルサービスをご利用いただけない場合は…

	電話	ファックス
東日本相談室→	043 - 351 - 1822	043 - 299 - 8280
西日本相談室→	06 - 6792 - 1583	06 - 6792 - 5993

受付時間　●月曜～土曜:9:00～18:00　●日曜・祝日:9:00～17:00（年末年始を除く）

修理のご相談など

【修理相談センター】（沖縄・奄美地区を除く）

0570 - 02 - 4649

全国どこからでも一律料金でご利用いただけます。
携帯電話からもご利用いただけます。

■〈PHS・IP電話やファクシミリをご利用〉または〈沖縄・奄美地区の方〉は…

	PHS／IP電話	ファックス
東日本地区→	043 - 299 - 3863	043 - 299 - 3865
西日本地区→	06 - 6792 - 5511	06 - 6792 - 3221
沖縄・奄美地区→	「那覇サービスセンター」098 - 861 - 0866（月～金 9:00～17:40）	

受付時間　●月曜～土曜:9:00～20:00　●日曜・祝日:9:00～18:00（年末年始を除く）

●電話番号・受付時間などについては、変更になることがあります。(2008.12)

シャープ株式会社

本　社　〒545-8522　大阪市阿倍野区長池町22番22号
情報システム事業本部　〒639-1186　奈良県大和郡山市美濃庄町492

PRINTED IN CHINA
08MKS (TINSJ1467EHZZ)
0GS9215380////